Τεχνολογικό Εκπαιδευτικό Ίδρυμα Πελοποννήσου

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Δημιουργία Εφαρμογής Μηχανογράφησης Για Super Market

Πτυχιακή Εργασία

Του

ΚΟΝΤΟΝΙΚΟΛΑΟΥ ΓΕΩΡΓΙΟΥ

Επιβλέπων: ΙΩΑΝΝΗΣ ΚΟΥΡΕΤΑΣ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχ ία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

Ημερομηνία (Ημέρα –Μήνας –Έτος):

Περίληψη

Ένα σύστημα μηχανογράφησης supermarket αποτελείτε από ένα σύνολο από πίνακες δεδομένων πχ προϊόντα προμηθευτές παραγγελίες κτλ. οπού εξυπηρετούν τις ανάγκες της επιχείρησης σε συνδυασμό με ένα κατάλληλο διαμορφωμένο περιβάλλον λογισμικού για προβολή ,επεξεργασία αυτών των δεδομένων με σκοπό την ομαλότερη λειτουργία της επιχείρησης .

Στόχος της πτυχιακής εργασίας είναι η αναπτύξει ενός συστήματος μηχανογράφησης σουπερμάρκετ το οποίο (i) θα επιτρέπει στον χρήστη να καταχωρεί επεξεργάζεται αλλά και να διαγραφεί τα δεδομένα του σχετικά με πχ (προϊόντα προμηθευτές παραγγελίες πωλήσεις) (ιι) Αυτόματη ενημέρωση βάσης για αγορές – πώλησης ,τιμές.

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή μου κ. Ιωάννη Κουρέτα για την επίβλεψη αυτής της πτυχιακής εργασίας και την ευκαιρία μου δόθηκε μέσω του συγκεκριμένου θέματος να αποκτήσω εμπειρία σε έναν ενδιαφέρον τομέα. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την υποστήριξη τους όλα αυτά τα χρόνια.

Περιεχόμενα

Κεφάλαιο 1
Ανάπτυξη λογισμικού

1.1 Σκοπός ανάπτυξης ενός λογισμικού

Στο κεφάλαιο αυτό θα αναλύσουμε τον τρόπο με τον οποίο δημιουργείτε ένα λογισμικό. Στης μέρες μας οι ηλεκτρονικοί υπολογιστές είναι ένα απαραίτητο εργαλείο διευκόλυνσης της δουλείας μας .Αφού μπορούν να μας εξοικονομήσουν χρόνο και χώρο. Αυτό επιτυγχάνεται μέσα από κατάλληλα σχεδιασμένα προγράμματα που στοχεύουν στο πρόβλημα μας. Όμως πως δημιουργούνται αυτά τα προγράμματά ? Ας υποθέσουμε για της ανάγκες της πτυχιακής ότι έχουμε μια επιχείρηση σουπερμάρκετ. Χωρίς κάποιο κατάλληλο λογισμικό θα έπρεπε να είχαμε βιβλιοθήκες γεμάτες έγγραφα που αφορούν την λειτουργικότητα της επιχείρησης μας . Αυτό θα είχε σαν συνέπεια τεράστιο όγκο δεδομένων σε υλική μορφή αλλά και φύρα χρόνου στην εύρεση ενός επιθυμητού δεδομένου. Με βάση ότι έχουμε αναφέρει μέχρι στιγμής έχουμε δημιουργήσει την ανάγκη **σύλληψης** μιας ιδέας για την δημιουργίας ενός λογισμικού με σκοπό την επίλυση του προβλήματός μας. Έχοντας μια βάση για το τι επιζητάμε από το λογισμικό παίρναμε στο στάδιο της **κατασκευής** της ιδέας . Τέλος λαμβάνουμε υπόψη την **χρήση/συντήρηση** αλλά και την **απόσυρση.** Η διαδικασία αυτή ονομάζεται φάσεις αναπτύξεις του κύκλου ζωής του λογισμικού. Παρακάτω φαίνεται το σχήμα των φάσεων.

Σχήμα 1.1 γενικές φάσεις του κύκλου ζωής του λογισμικού

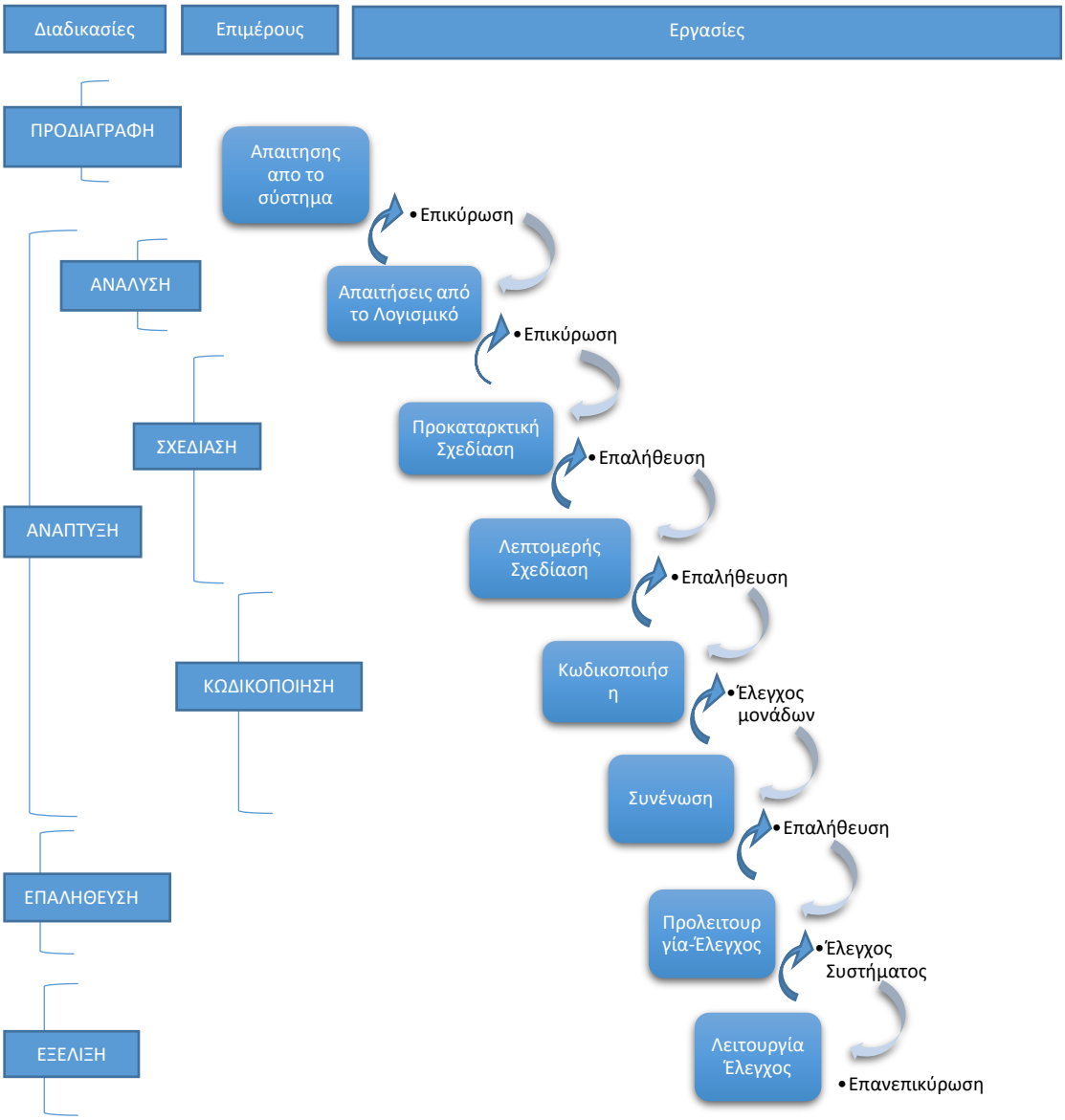
1.2 Ανάπτυξη λογισμικού

Στην τεχνολογία λογισμικού, η μεθοδολογία ανάπτυξης λογισμικού (ή διαδικασία ανάπτυξης λογισμικού ή ανάπτυξης εφαρμογών) είναι μία διαίρεση της ανάπτυξης λογισμικού σε διακριτές φάσεις (ή στάδια) με σκοπό τον καλύτερο σχεδιασμό και διαχείριση. Αυτό επιτυγχάνεται με τα μοντέλα κύκλου ζωής λογισμικού. Παράδειγμα μοντέλων :

- Μοντέλο του καταρράκτη
- Μοντέλο προτυποποίησης
- Μοντέλο λειτουργικής επαύξησης
- Σπειροειδές μοντέλο
- Μοντέλο πίδακα
- Γενικό μοντέλο

1.2.1 Μοντέλο του καταρράκτη

Ένα από τα πιο διαδεδομένα μοντέλα κύκλου ζωής είναι αυτό του καταρράκτη (Waterfall), το οποίο φαίνεται στο Σχήμα 1.2.1. Η κεντρική ιδέα του μοντέλου του καταρράκτη είναι ότι το σύστημα λογισμικού αναπτύσσεται περνώντας ολόκληρο από διαδοχικές επιμέρους φάσεις, καθεμία από τις οποίες θεωρείται περατωμένη με την παραγωγή ορισμένων συστατικών λογισμικού. Κάθε επιμέρους φάση ολοκληρώνεται με μια εργασία επαλήθευσης / επικύρωσης των προϊόντων της, κατά την οποία αποφασίζεται η μετάβαση ή όχι στην επόμενη. Το λογισμικό εμφανίζεται πλήρες, δηλαδή με όλα τα λειτουργικά του χαρακτηριστικά, από την επιμέρους φάση της συνένωσης και μετά. Χαρακτηριστικό του μοντέλου του καταρράκτη είναι ότι, για να ξεκινήσει μια φάση πρέπει να έχει ολοκληρωθεί πλήρως η προηγούμενη. Η ανάπτυξη με τον τρόπο αυτό χαρακτηρίζεται ακολουθιακή, διότι οι επιμέρους φάσεις από τις οποίες διέρχεται είναι διακριτές και ακολουθούν η μία την άλλη.


Διαδικασίες
Επιμέρους
Εργασίες
ΠΡΟΔΙΑΓΡΑΦΗ
Απαιτησης απο το σύστημα
Επικύρωση
ΑΝΑΛΥΣΗ
Απαιτήσεις από το Λογισμικό
Επικύρωση
ΣΧΕΔΙΑΣΗ
Προκαταρκτική Σχεδίαση
Επαλήθευση
ΑΝΑΠΤΥΞΗ
Λεπτομερής Σχεδίαση
Επαλήθευση
ΚΩΔΙΚΟΠΟΙΗΣΗ
Κωδικοποιήση
Έλεγχος μονάδων
Συνένωση
Επαλήθευση
ΕΠΑΛΗΘΕΥΣΗ
Προλειτουργία-Έλεγχος
Έλεγχος Συστήματος
ΕΞΕΛΙΞΗ
Λειτουργία Έλεγχος
Επανεπικύρωση

Σχήμα 1.2.1

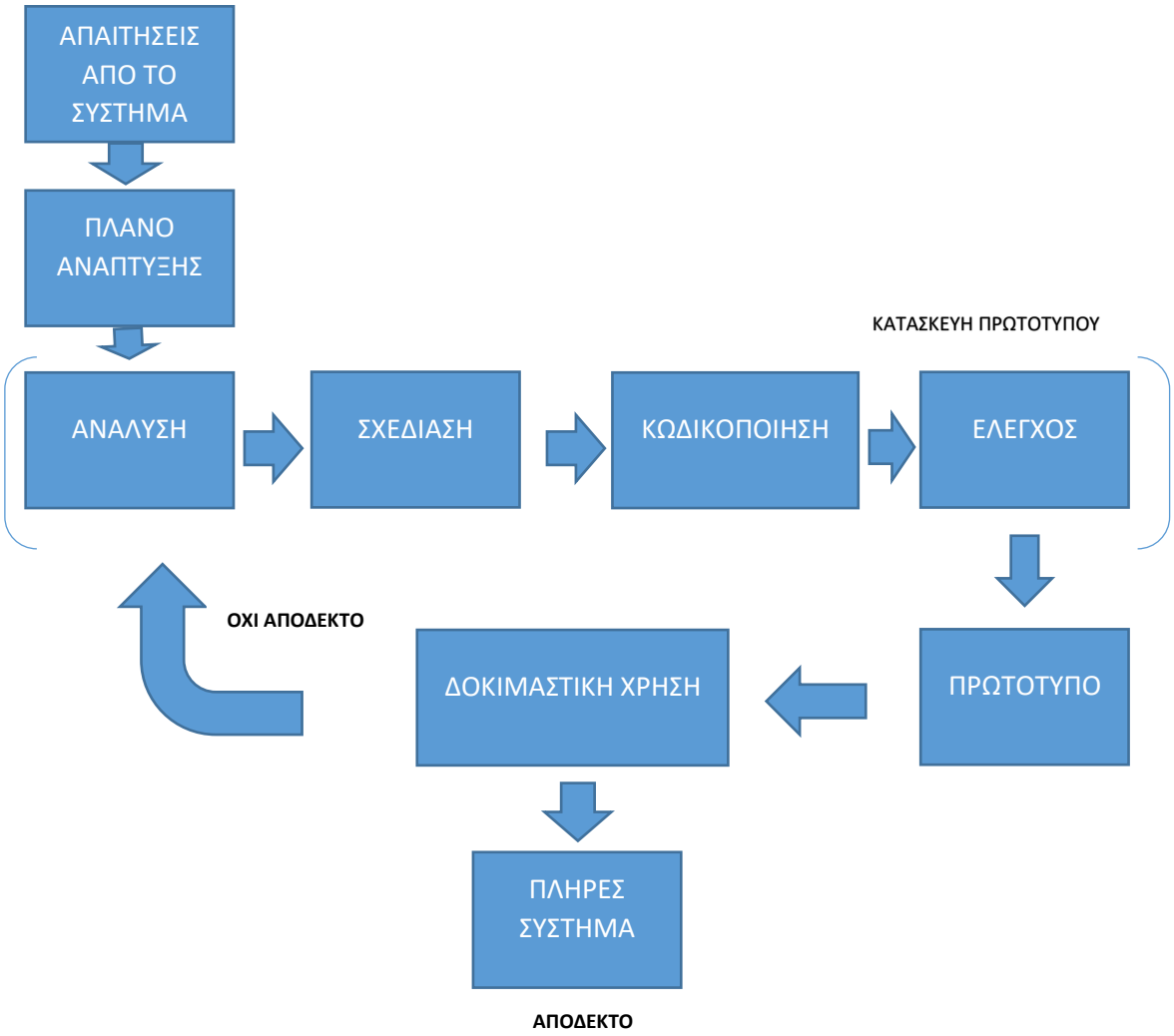
Αρχικά καθορίζονται **οι απαιτήσεις από το σύστημα** και το **λογισμικό**, αντίστοιχα. Ακολούθως γίνεται η προκαταρκτική και η λεπτομερής σχεδίαση του λογισμικού, αντίστοιχα. Κατά την **προκαταρκτική σχεδίαση** καθορίζονται οι μονάδες που θα αποτελούν το λογισμικό, καθώς και οι συσχετίσεις μμεταξύ τους. Ο καθορισμός αυτός μπορεί να γίνει σε περισσότερα από ένα επίπεδα λεπτομέρειας, ανάλογα με το μμέγεθος και την πολυπλοκότητα του λογισμικού. Το πρώτο επίπεδο (αυτό με τη μικρότερη λεπτομέρεια) περιέχει τα υποσυστήματα, το δεύτερο περιέχει τις μονάδες μέσα σε κάθε υποσύστημά Κ.Ο.Κ. Κατά τη **λεπτομερή σχεδίαση** καθορίζεται η εσωτερική δομή κάθε μονάδας λογισμικού, η οποία αντιστοιχεί πρακτικά σε μμονάδες πηγαίου κώδικα προγράμματος. Ο καθορισμός αυτός περιλαμβάνει όλα τα απαραίτητα στοιχεία (αλγόριθμοι, δομές δεδομένων κτλ.), ώστε η **συγγραφή του πηγαίου κώδικα**, που ακολουθεί, να είναι μια διαδικασία
διεκπεραίωσης και μόνο. Ακολουθεί η **συνένωση** των μονάδων σε σύστημα και ο έλεγχος του συστήματος, η ολοκλήρωση του οποίου επιτρέπει την παράδοση ολόκληρου του προϊόντος στον πελάτη και το πέρασα στη φάση της **λειτουργίας και συντήρησης**..

1.2.2 Μοντέλο προτυποποίησης

Η κεντρική ιδέα του *μοντέλου προτυποποίησης* είναι η ανάπτυξη του λογισμικού όχι εξολοκλήρου, αλλά σε τμήματα, που ονομάζονται «πρωτότυπα». Οι διαδικασίες ανάπτυξης επαναλαμβάνονται για ένα τμήμα του συστήματος κάθε φορά και, για το λόγο αυτό, το μοντέλο χαρακτηρίζεται ως *επαναληπτικό*. Κάθε πρωτότυπο περιλαμβάνει τις βασικές από τις λειτουργίες που προορίζεται να εκτελεί το λογισμικό και τίθεται σε δοκιμασία από τον πελάτη. Από εκεί συλλέγονται παρατηρήσεις και η διαδικασία κατασκευής νέου πρωτοτύπου επαναλαμβάνεται μέχρις ότου ένα πρωτότυπο να ικανοποιεί τις απαιτήσεις, δηλαδή να εκτελεί τις επιθυμητές λειτουργίες του λογισμικού με τρόπο ικανοποιητικό και να γίνεται αποδεκτό από τον πελάτη (Σχήμα 1.2.2). Από το σημείο αυτό και μετά μπορούν να προστεθούν και οι υπόλοιπες λειτουργίες, ώστε το λογισμικό να ολοκληρωθεί.

Ένα σημαντικό πλεονέκτημα του μοντέλου αυτού είναι η δυνατότητα απόκτησης άποψης για την εφαρμογή λογισμικού νωρίτερα από ό,τι στο μοντέλο του καταρράκτη. Αυτό μπορεί να γλιτώσει την ανάπτυξη από καθυστερήσεις (και συνεπαγόμενα κόστη) ή ακόμη και από ολική αποτυχία, τα οποία θα επέρχονταν, αν ο κατασκευαστής αναγκαζόταν να οπισθοδρομήσει την ανάπτυξη, ενώ αυτή είχε προχωρήσει πολύ. Παράλληλα, ιδιαίτερη σημασία αποκτά η διοίκηση του έργου, η οποία πρέπει να εξασφαλίζει την υλοποιησιμότητα του πρωτοτύπου και την εύκολη τροποποίησή του. Κάθε κατασκευή πρωτοτύπου μπορεί να θεωρηθεί ως ένα μικρό έργο λογισμικού το οποίο κατασκευάζεται με διαδικασίες που μπορούν να ακολουθούν άλλα μοντέλα κύκλου ζωής, όπως αυτό του καταρράκτη.

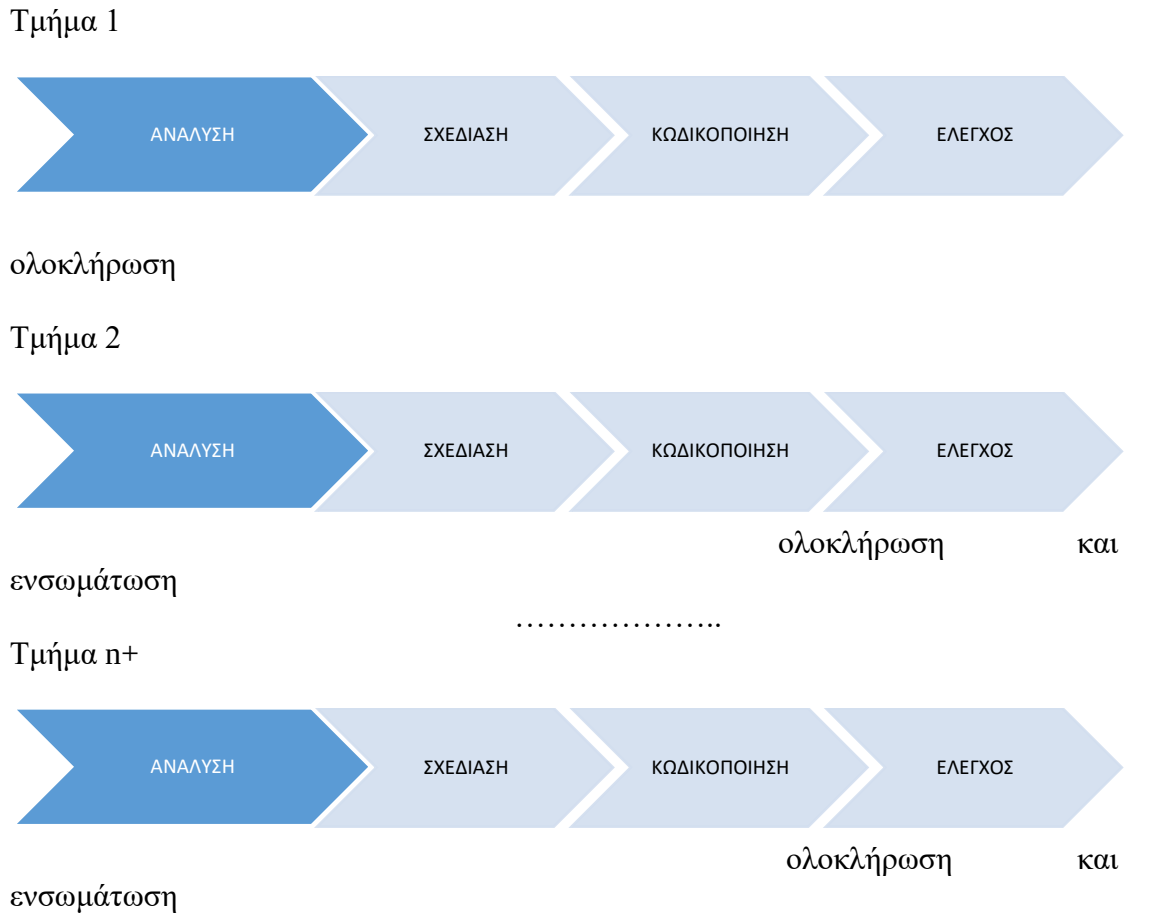
Με βάση τις παραπάνω παρατηρήσεις, το μοντέλο προτυποποίησης χρησιμοποιείται στην ανάπτυξη εφαρμογών λογισμικού για τις απαιτήσεις από τις οποίες δεν υπάρχει βεβαιότητα στην αρχή της ανάπτυξης, οπότε δεν μπορούν να συμφωνηθούν και να παγιοποιηθούν. Τέτοιες είναι εφαρμογές που κατασκευάζονται για πρώτη φορά ή που είναι στενά εξαρτημένες από τον πελάτη, χωρίς να υπάρχει αποδεκτό προηγούμενο παράδειγμα. Ωστόσο, το μέγεθος των εφαρμογών αυτών δεν μπορεί να είναι ιδιαίτερα μεγάλο, διότι ο χρόνος ανάπτυξης κάθε πρωτοτύπου μεγαλώνει και η απαιτούμενη ευελιξία μειώνεται


ΑΠΑΙΤΗΣΕΙΣ ΑΠΟ ΤΟ ΣΥΣΤΗΜΑ
ΠΛΑΝΟ ΑΝΑΠΤΥΞΗΣ
ΚΑΤΑΣΚΕΥΗ ΠΡΩΤΟΤΥΠΟΥ
ΑΝΑΛΥΣΗ
ΣΧΕΔΙΑΣΗ
ΚΩΔΙΚΟΠΟΙΗΣΗ
ΕΛΕΓΧΟΣ
ΟΧΙ ΑΠΟΔΕΚΤΟ
ΔΟΚΙΜΑΣΤΙΚΗ ΧΡΗΣΗ
ΠΡΩΤΟΤΥΠΟ
ΠΛΗΡΕΣ ΣΥΣΤΗΜΑ
ΑΠΟΔΕΚΤΟ

Σχήμα 1.2.2

1.2.3 Μοντέλο λειτουργικής επαύξησης

Το *μοντέλο της λειτουργικής επαύξησης* συνδυάζει την ακολουθιακή ανάπτυξη του μοντέλου του καταρράκτη με την τμηματική ανάπτυξη του μοντέλου της προτυποποίησης. Κεντρική ιδέα είναι η κατάτμηση του υπό κατασκευή λογισμικού σε τμήματα που αναπτύσσονται ανεξάρτητα, ακολουθώντας το καθένα ακολουθιακή ανάπτυξη σύμφωνα με το μοντέλο του καταρράκτη, όπως φαίνεται στο Σχήμα 1.2.3 . Κατά την αρχική φάση ανάλυσης και σχεδίασης αποφασίζονται τα τμήματα στα οποία θα κατανεμηθεί η εφαρμογή, η ανάπτυξη των οποίων γίνεται στη συνέχεια ανεξάρτητα και παράλληλα. Όταν ολοκληρώνεται η ανάπτυξη κάθε τμήματος, αυτό ενσωματώνεται στο σύνολο της εφαρμογής. Το μοντέλο της λειτουργικής επαύξησης χρησιμοποιείται στην ανάπτυξη μεγάλων εφαρμογών λογισμικού για τις οποίες ισχύουν οι απαιτήσεις του μοντέλου του καταρράκτη, δηλαδή σαφής γνώση και μικρή ή καθόλου μεταβλητότητα των απαιτήσεων κατά την ανάπτυξη.



Σχήμα 1.2.3

1.2.4 Σπειροειδές μοντέλο

Πρόκειται για μια γενίκευση των μμοντέλων της λειτουργικής επαύξησης και της προτυποποίησης, με σημαντικά νέα στοιχεία: η Οι φάσεις και οι διαδικασίες ανάπτυξης λογισμικού δεν είναι προκαθορισμένες από το μοντέλο, αλλά εξειδικεύονται στο χώρο της εφαρμογής του. Η ανάπτυξη ολόκληρου του συστήματος χωρίζεται σε πολλούς κύκλους, σε καθέναν από τους οποίους προστίθενται νέα λειτουργικά χαρακτηριστικά στο σύστημα. Πριν από την έναρξη κάθε κύκλου γίνεται μια μελέτη σκοπιμότητας και ανάλυση κινδύνων, από την οποία προκύπτουν, αφενός, οι συγκεκριμένες εργασίες που θα εκτελεστούν μέσα στον κύκλο, αφετέρου, η ίδια η εφικτότατα εκτέλεσης του κύκλου αυτού. Στο σπειροειδές μοντέλο στο Σχήμα 1.2.4 διακρίνονται τέσσερις κατηγορίες εργασιών:
προσδιορισμός στόχων, **εντοπισμός και επίλυση κινδύνων**, **εκτέλεση διαδικασιών ανάπτυξης και επαλήθευση**, καθώς και **εργασίες προγραμματισμού**.

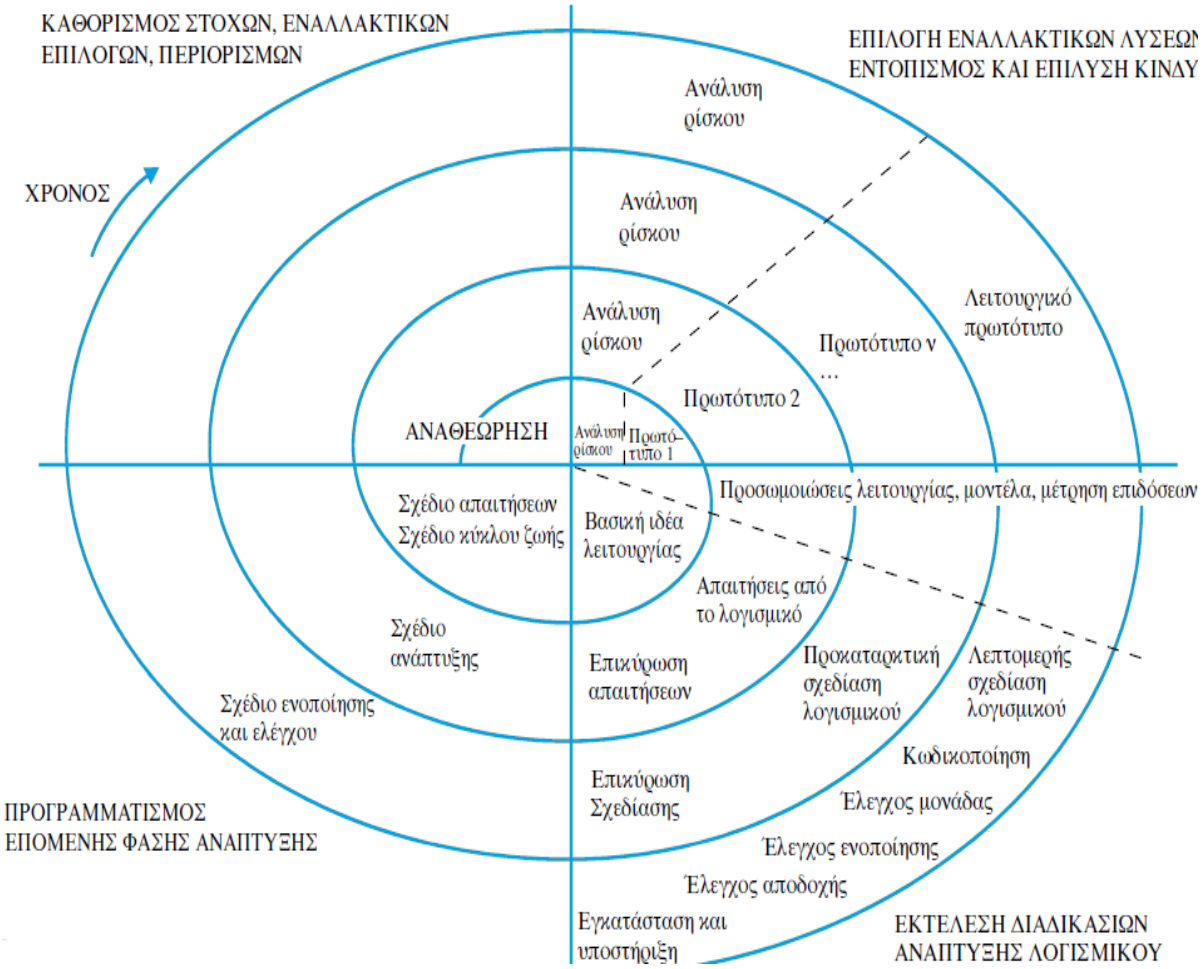
- **προσδιορισμό στόχων** καθορίζονται τα αντικείμενα εργασιών κάθε επανάληψης, καταγράφονται οι περιορισμοί επί του προϊόντος, αλλά και επί της διαδικασίας για την οποία κατασκευάζεται ένα αναλυτικό πλάνο διοίκησης. Επίσης, καταγράφονται οι κίνδυνοι που εμπεριέχει η διαδικασία και οι εναλλακτικές λύσεις, όπου υπάρχουν.

- **επίλυσης κινδύνων** αναλύονται οι κίνδυνοι που έχουν καταγραφεί και αποτιμάται κάθε εναλλακτική λύση. Στο σημείο αυτό λαμβάνονται αποφάσεις για τη συνέχιση ή όχι της ανάπτυξης, για το μοντέλο που θα ακολουθηθεί στη συγκεκριμένη επανάληψη, για την κατασκευή ή όχι πρωτοτύπου κ.ά.

- **εκτέλεση** των βημάτων της **διαδικασίας ανάπτυξης** λογισμικού που έχει επιλεγεί για το τμήμα εκείνο του συστήματος που αφορά η τρέχουσα επανάληψη.

- Τέλος, Μετά την επαλήθευση των αποτελεσμάτων . ενδιάμεσων προϊόντων λογισμικού γίνεται **προγραμματισμός** της συνέχισης της ανάπτυξης.

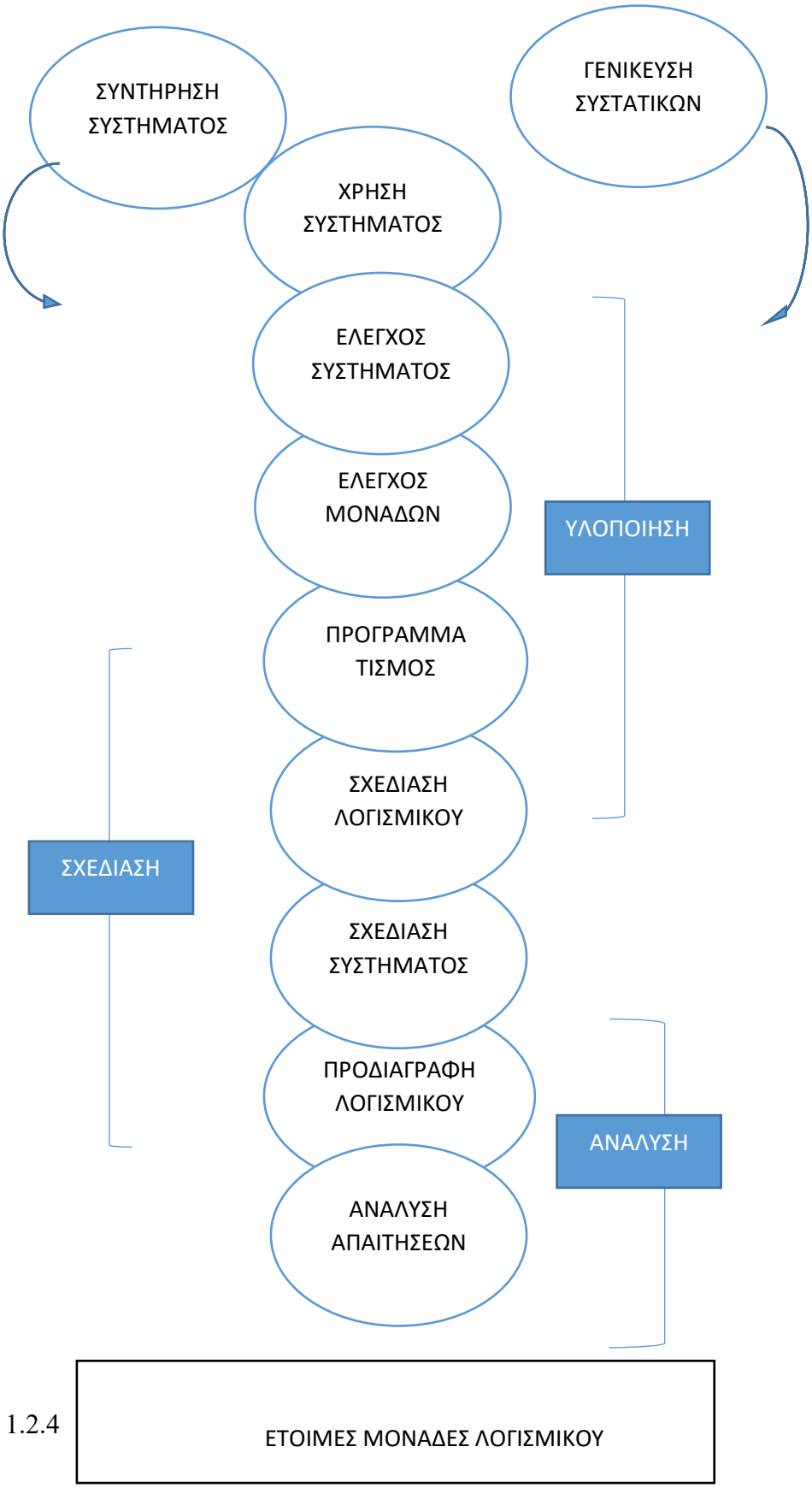
Το σπειροειδές μμοντέλο δεν καθορίζει εκ των προτέρων ποιες ακριβώς είναι οι εργασίες ανάπτυξης λογισμικού που πρέπει να γίνουν ούτε σε ποια έκταση του συστήματος αυτές θα εφαρμοστούν. Διαφορετικές διαδικασίες ανάπτυξης μπορεί να επιλεγούν για διαφορετικά τμήματα του λογισμικού. Αυτό που προτείνει είναι ότι *ο καθορισμός των λεπτομέρειών υλοποίησης πρέπει να γίνεται συνεχώς κατά την ανάπτυξη*


ΚΑΘΟΡΙΣΜΟΣ ΣΤΟΧΩΝ, ΕΝΑΛΛΑΚΤΙΚΩΝ
ΕΠΙΛΟΓΩΝ, ΠΕΡΙΟΡΙΣΜΩΝ
ΕΠΙΛΟΓΗ ΕΝΑΛΛΑΚΤΙΚΩΝ ΛΥΣΕΩΝ
ΕΝΤΟΠΙΣΜΟΣ ΚΑΙ ΕΠΙΛΥΣΗ ΚΙΝΔΥ
ΧΡΟΝΟΣ
Ανάλυση ρίσκου
Ανάλυση ρίσκου
Ανάλυση ρίσκου
Λειτουργικό πρωτότυπο
Πρωτότυπο ν
...
Πρωτότυπο 2
ΑΝΑΘΕΩΡΗΣΗ
Ανάλυση ρίσκου
Πρωτό-τυπο 1
Προσομοιώσεις λειτουργίας, μοντέλα, μέτρηση επιδόσεων
Σχέδιο απαιτήσεων
Σχέδιο κύκλου ζωής
Βασική ιδέα λειτουργίας
Απαιτήσεις από το λογισμικό
Σχέδιο ανάπτυξης
Επικύρωση απαιτήσεων
Προκαταρκτική σχεδίαση λογισμικού
Λεπτομερής σχεδίαση λογισμικού
Σχέδιο ενοποίησης και ελέγχου
Κωδικοποίηση
Επικύρωση Σχεδίασης
Έλεγχος μονάδας
Έλεγχος ενοποίησης
Έλεγχος αποδοχής
ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
ΕΠΟΜΕΝΗΣ ΦΑΣΗΣ ΑΝΑΠΤΥΞΗΣ
Εγκατάσταση και υποστήριξη
ΕΚΤΕΛΕΣΗ ΔΙΑΔΙΚΑΣΙΩΝ
ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ

Σχήμα 1.2.4

1.2.5 Μοντέλο πίδακα

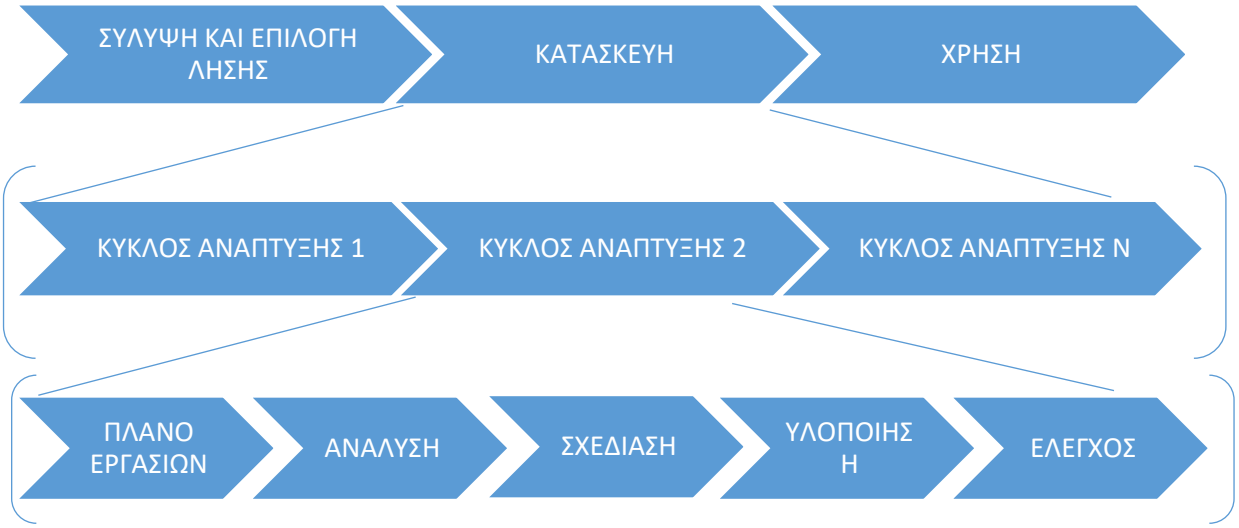
Οι πρώτες προσεγγίσεις του θέματος με βάση την *αντικειμενοστραφή* τεχνολογία βασιζόμενες σε δύο ιδιαίτερα γνωρίσματά της: πρώτον, ότι οι έννοιες «ανάλυση σχεδίαση. κωδικοποίηση» έρχονται στο αντικειμενοστραφές παράδειγμα πολύ πιο κοντά και, δεύτερον, ότι το αποτέλεσμα κάθε διαδικασίας κατασκευής λογισμικού είναι όχι μόνο ένα σύστημα, αλλά και επαναχρησιμοποιήσιμες μονάδες, οι οποίες μπορούν να χρησιμοποιηθούν από τις πρώτες φάσεις της ανάπτυξης μελλοντικών συστημάτων. Με τον τρόπο αυτό προέκυψε το *μοντέλο του πίδακα*, που φαίνεται στο Σχήμα 1.2.5 Κατά την ανάπτυξη παρατηρούνται επικαλύψεις των φάσεων «ανάλυση . σχεδίαση . κωδικοποίηση», οι οποίες φαίνονται με την επικάλυψη των κύκλων στο σχήμα. Κατά το τέλος της ανάπτυξης, ορισμένα από τα συστατικά λογισμικού που έχουν παραχθεί ενσωματώνονται σε μια «δεξαμενή» συστατικών και διατίθενται για να χρησιμοποιηθούν στην ανάπτυξη και νέων συστημάτων.


ΣΥΝΤΗΡΗΣΗ ΣΥΣΤΗΜΑΤΟΣ
ΓΕΝΙΚΕΥΣΗ ΣΥΣΤΑΤΙΚΩΝ
ΧΡΗΣΗ ΣΥΣΤΗΜΑΤΟΣ
ΕΛΕΓΧΟΣ ΣΥΣΤΗΜΑΤΟΣ
ΕΛΕΓΧΟΣ ΜΟΝΑΔΩΝ
ΥΛΟΠΟΙΗΣΗ
ΠΡΟΓΡΑΜΜΑ ΤΙΣΜΟΣ
ΣΧΕΔΙΑΣΗ ΛΟΓΙΣΜΙΚΟΥ
ΣΧΕΔΙΑΣΗ
ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΟΣ
ΠΡΟΔΙΑΓΡΑΦΗ ΛΟΓΙΣΜΙΚΟΥ
ΑΝΑΛΥΣΗ
ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ
ΕΤΟΙΜΕΣ ΜΟΝΑΔΕΣ ΛΟΓΙΣΜΙΚΟΥ

Σχήμα 1.2.4

1.2.6 Γενικό μοντέλο

Μια περιγραφή ενός σύγχρονου μοντέλου κύκλου ζωής λογισμικού περιέχει μόνο γενικές κατευθύνσεις, οι οποίες εξειδικεύονται στο εκάστοτε περιβάλλον ανάπτυξης, πρόβλημα κτλ. Επίσης, δεν είναι άρρηκτα συνδεδεμένο με κάποια μμεθοδολογία ανάπτυξης λογισμικού, αλλά μμπορεί να εξειδικευτεί για την πρακτική τού κάθε κατασκευαστή. Ένα τέτοιο μμοντέλο φαίνεται στο Σχήμα 1.2.6 και μπορεί να χαρακτηριστεί ως απόγονος πολλών από τα μοντέλα που αναλυθήκαν. Το γενικό πλαίσιο του μοντέλου αυτού περιλαμβάνει τις φάσεις σύλληψης, κατασκευής και λειτουργίας. Καθεμιά από αυτές αναλύεται σε επιμέρους εργασίες, σύμφωνα με τα χαρακτηριστικά του εκάστοτε περιβάλλοντος. Ιδιαίτερα η γενική φάση της κατασκευής αναλύεται σε «κύκλους ανάπτυξης», καθένας εκ των οποίων προσθέτει νέα χαρακτηριστικά και λειτουργίες στο υπό κατασκευή λογισμικό. Τα επιμέρους βήματα μέσα σε κάθε κύκλο ανάπτυξης μοιάζουν με τα βήματα του μοντέλου του καταρράκτη, μόνο που δεν εφαρμόζονται για ολόκληρο το σύστημα, αλλά για το μικρό μέρος του που κατασκευάζεται στον εν λόγω κύκλο, όπως στο μμοντέλο της προτυποποίησης. Για την εκκίνηση κάθε κύκλου ανάπτυξης μμπορεί να έχει προηγηθεί ανάλυση ρίσκου και σκοπιμότητας, όπως στο σπειροειδές μοντέλο.



Σχήμα 1.2.6

1.2.7) Πίνακας σύγκριση μοντέλων κύκλου ζωής λογισμικού

Μοντέλο	**Μέγεθος εφαρμογών**	**Μεταβολές στις απαίτησης**	**Προσαρμοστικότητα στον κατασκευαστή**	**Διάδοση**
Καταρράκτη	Μικρό έως μεσαίο	Ανεπιθύμητες	Καμία	Μεγάλη με τάση μείωσης
Προτυποποίησης	Μικρό έως μεσαίο	Δεκτές	Μικρή	Μικρή με τάση αύξησης
Λειτουργικής επαύξησης	Μεσαίο έως μεγάλο	Ανεπιθύμητες	Καμία	Μικρή με τάση μείωσης
Σπειροειδές	Μεσαίο έως μεγάλο	Δεκτές	Αρκετή	Μικρή με τάση μείωσης
πίδακα	Οποιαδήποτε	Δεκτές	Αρκετή	Μικρή
Γενικό	Οποιαδήποτε	Δεκτές	Μεγάλη	Μικρή με ισχυρές τάσεις αυξήσεις

Κεφάλαιο 2
Σχεδίαση λογισμικού

2.1 Σκοπός της Σχεδίασης λογισμικού

Στο κεφάλαιο αυτό θα αναλύσουμε το αντικείμενο της σχεδίασης του λογισμικού. Γι' αυτό χρειαζόμαστε ένα τρόπο περιγραφής της κατασκευής του λογισμικού ούτος ώστε να πλήρη της προδιάγραφες που έχουν τεθεί. Για να μπορέσουμε να ξεκινήσουμε την σχεδίαση είναι απαραίτητη η ύπαρξη των προδιαγραφών. Οι παραπάνω αναφορές που παράγονται κατά την σχεδίαση ονομάζονται <<σχέδιο λογισμικού>>.

Σχέδιο λογισμικού

- Είναι η περιγραφή των μονάδων που αποτελούν το λογισμικό των συσχετίσεων μεταξύ τους ,της διάταξης τους, καθώς και της εσωτερικής τος λεπτομέρειας

Σκοπός της σχεδίασης είναι να δοθεί η καλύτερη δυνατή λύση στις εκάστοτε συνθήκες.

2.2 Τεχνοτροπίες σχεδίασης

Το πρόβλημα της σχεδίασης ενός λογισμικού λόγο της δυσκολίας του μπορεί να αντιμετωπιστεί μέσο των στρατηγικών προσεγγίσεων .Αυτές η μεθοδολογίες κατατάσσονται σε δυο μεγάλες κατηγορίες :

- τις προσανατολισμένες στις διαδικασίες
- τις προσανατολισμένες στα αντικείμενα

2.2.1 Δομημένη σχεδίαση

Οι μεθοδολογίες που ακολουθούν αυτή την προσέγγιση προτείνουν τρόπους αποσύνθεσης του συστήματος *από πάνω προς τα κάτω* σε μια ιεραρχία διαδικασιών, συναρτήσεων και άλλων ενεργών μονάδων λογισμικού. Όσο κατεβαίνει κανείς στην ιεραρχία αυτή, τόσο μεγαλύτερη λεπτομέρεια συναντά, μέχρις ότου φτάσει στις απλές δομικές μονάδες, δηλαδή τις εντολές της γλώσσας προγραμματίσου.

2.2.2 Αντικειμενοστραφής σχεδίαση

Η *αντικειμενοστραφής προσέγγιση* ακολουθεί ένα διαφορετικό δρόμο: αντί το σύστημα να θεωρείται ως μια ιεραρχία διαδικασιών, ανεξάρτητων από τα δεδομένα, θεωρείται ως μια συλλογή οντοτήτων, καθεμία εκ των οποίων περικλείει και διαδικασίες και δεδομένα. Η προσέγγιση βασίζεται στην ιδέα ότι στον πραγματικό κόσμο δεδομένα και διαδικασίες μπορούν να ιδωθούν ενιαία με βάση το πεδίο ευθύνης κάποιων οντοτήτων που ονομάζονται «αντικείμενα». Κάθε αντικείμενο παρέχει στο περιβάλλον του ένα σύνολο υπηρεσιών της ευθύνης του. Η συνεργασία του συνόλου των αντικειμένων του πεδίου μιας εφαρμογής λογισμικού παράγει το επιθυμητό αποτέλεσμα.

2.3 Επίπεδα σχεδίου λογισμικού

Στην ενότητα 2.1 δώσαμε έναν ορισμό για το σχέδιο λογισμικού ως <<Μια περιγραφή των μονάδων που αποτελούν το λογισμικό των συσχετίσεων μεταξύ τους ,της διάταξης τους, καθώς και της εσωτερικής τος λεπτομέρειας >> Προκειμένου να γίνει δυνατή η περιγραφή αυτή, πρέπει να αντιμετωπίσουμε το πρόβλημα σε τέσσερα επίπεδα :

- **Αρχιτεκτονική σχεδίαση:** Αφορά τον καθορισμό τού ποιες είναι οι μμονάδες που συγκροτούν το σύστημα λογισμικού και πώς αυτές ανατίθενται για εκτέλεση (διατάσσονται) στις υπολογιστικές μμονάδες που είναι διαθέσιμες.

- **Σχεδίαση διαπροσωπικών:** Αφορά τον καθορισμό της επικοινωνίας των μονάδων μεταξύ τους, με άλλα συστήματα λογισμικού, με άλλες συσκευές, καθώς και με τον άνθρωπο. Λέγοντας «καθορισμός επικοινωνίας» εννοούμε την περιγραφή του ποια μονάδα επικοινωνεί με ποια, με ποιες παραμέτρους, καθώς και με όποιο άλλο στοιχείο απαιτείται για να περιγράφει επαρκώς, κατά περίπτωση, η επικοινωνία.

- **Λεπτομερής σχεδίαση μμονάδων:** Αφορά τον καθορισμό της εσωτερικής δομής κάθε μμονάδας λογισμικού προκειμένου αυτή να ικανοποιεί, αφενός, τις λειτουργικές απαιτήσεις και, αφετέρου, να συνεργάζεται με τις άλλες μονάδες όπως έχει καθοριστεί κατά την αρχιτεκτονική και τη σχεδίαση δεδομένων.

- **Σχεδίαση δεδομένων:** Πρόκειται για τη λεπτομερή σχεδίαση των δεδομένων, η τήρηση των οποίων αποτελεί απαίτηση από το λογισμικό, η οποία έχει τεθεί στη φάση προδιαγραφής των απαιτήσεων.

Κεφάλαιο 3
Εργαλεία ανάπτυξης λογισμικού

3.1 Περίληψη

Έχοντας ανάλυση στο προηγούμενο κεφάλαιο την δομή του λογισμικού σαν μια ιδέα περνάμε πλέον στο να το υλοποιήσουμε και στην πράξη .Σε αυτό το σημείο έχουμε να αναρωτηθούμε πιο εργαλείο αναπτύξεις θα χρησιμοποιήσουμε για να επιτευχθεί ο στόχος μας με γνώμονα τις απαίτησης των εργαλείων ως προς το σύστημα .με βάση τα όσα αναφερθήκαν παραπάνω το κεφάλαιο αυτό χωρίζεται σε δυο ενότητες:

- **Εύρεση κατάλληλων εργαλείων ανάπτυξης ως προς το λογισμικό**
- **Απαίτησης εργαλείων ανάπτυξης λογισμικού ως προς το σύστημα**

3.2 Εύρεση κατάλληλων εργαλείων ανάπτυξης ως προς το λογισμικό

Κατά την διάρκεια της σχεδίασης ενός λογισμικού έχουμε να αναρωτηθούμε τα μέσα με τα οποία θα επιτευχθεί ο στόχος μας . Στην παρούσα πτυχιακή με θέμα <<την αναπτύξει ενός ERP(Enterprise resource planning) συστήματος supermarket >> χρειαζόμαστε :

- Πλατφόρμα Βάσεων δεδομένων
- Πλατφόρμα ανάπτυξης λογισμικού

3.2.1 Πλατφόρμα Βάσεων δεδομένων

Τα προγράμματα για την δημιουργία βάσεων δεδομένων ποικίλουν. Τα ποιο διαδεδομένα είναι ο Oracle., MySQL και ο SQLserver. Αν αναλύσουμε όμως την κάθε επιλογή μας με βάση της ανάγκες μας τότε προκύπτει πια είναι η καταλληλότερη πλατφόρμα για μας .

Ο παρακάτω πίνακας δείχνει της δυνατότητες της κάθε πλατφόρμας.

Feature	Oracle	MySQL	SQL Server
Περιβάλλον	GUI, SQL	SQL	GUI, SQL, Various
Υποστηριζόμενες γλώσσες	Αρκετές όπως C, C#, C++, Java, Ruby, και Objective C	Αρκετές όπως C, C#, C++, D, Java, Ruby, και Objective C	Java, Ruby, Python, VB, .Net, και PHP
Λειτουργικό Σύστημα	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris	Windows
Άδεια	Proprietary	Open source	Proprietary

Στην παρούσα πτυχιακή επιλέξαμε τον sqlserver .

Λίγα λόγια για τον sqlserver :

Ο SQL Server είναι μια σχεσιακή βάση δεδομένων, η οποία αναπτύσσεται από τη Microsoft. Οι κύριες γλώσσες που χρησιμοποιούνται είναι η T-SQL και η ANSI SQL. Ο SQL Server βγήκε για πρώτη φορά στην αγορά το 1989 σε συνεργασία με την Sybase.

Η κύρια μονάδα αποθήκευσης στοιχείων είναι μια βάση δεδομένων, η οποία αποτελείται από μια συλλογή πινάκων και κώδικα.

Αποθήκευση

Η κεντρική βάση δεδομένων του SQL υποστηρίζει διαφορετικούς τύπους, συμπεριλαμβανομένων των ακεραίων αριθμών, αριθμών κινητής υποδιαστολής, δεκαδικών, αλφαριθμητικών, Varchar (σειρές χαρακτήρων μεταβλητού μήκους), δυαδικών αριθμών (για τα μη δομημένα δεδομένα), κειμένων (για κείμενα).Επιτρέπει επίσης καθορισμένους από το χρήστη σύνθετους τύπους δεδομένων (UDTs), δηλαδή τύπους που βασίζονται στους βασικούς τύπους αλλά μπορούν να τροποποιηθούν. Τα στοιχεία στη βάση δεδομένων αποθηκεύονται σε ένα (ή περισσότερα) αρχεία με επέκταση .mdf.

Τα δευτεροβάθμια στοιχεία αποθηκεύονται στο αρχείο με επέκταση .ndf. Το αρχείο καταγραφής το οποίο περιέχει όλες τις πρόσφατες αλλαγές στη βάση δεδομένων αποθηκεύεται σε αρχείο με επέκταση .ldf.Ο χώρος αποθήκευσης που διατίθεται σε μια βάση δεδομένων διαιρείται σε διαδοχικά αριθμημένες σελίδες, κάθε μία από τις οποίες έχει μέγεθος 8 KB.

Ενδιάμεση μνήμη

Οι σελίδες αποθηκεύονται στην ενδιάμεση (buffer) μνήμη RAM για να ελαχιστοποιηθεί η μεταφορά δεδομένων προς και από τον σκληρό δίσκο. Οποιαδήποτε σελίδα 8 KB μπορεί να είναι αποθηκευμένη στη μνήμη, και το σύνολο όλων των σελίδων που αποθηκεύονται σε μία περίοδο καλείται λανθάνουσα μνήμη (cache).Το ποσό μνήμης που είναι διαθέσιμο στον κεντρικό διακομιστή SQL αποφασίζει πόσες σελίδες θα εναποθηκευθούν στη λανθάνουσα μνήμη.

Ταυτοχρονισμός

Ο κεντρικός διακομιστής SQL παρέχει δύο τρόπους ελέγχου του ταυτοχρονισμού: απαισιόδοξος ταυτοχρονισμός και αισιόδοξος ταυτοχρονισμός.Όταν ο απαισιόδοξος έλεγχος ταυτοχρονισμού χρησιμοποιείται, ο κεντρικός διακομιστής SQL ελέγχει την ταυτόχρονη πρόσβαση με τη χρησιμοποίηση κλειδώματος (locks). Τα κλειδώματα μπορούν είτε να διαμοιράζονται είτε να είναι αποκλειστικά.

Ανάκτηση δεδομένων

Η ερώτηση (query) είναι ο κύριος τρόπος για την ανάκτηση στοιχείων από μια βάση δεδομένων. Η ερώτηση εκφράζεται χρησιμοποιώντας μια παραλλαγή της αποκαλούμενου SQL T-SQL, είναι μια διάλεκτος SQL που αναπτύχθηκε από την

Microsoft και Sybase. Η T-SQL είναι πολύ κοντά στα ANSI standards που έχουν καθιερωθεί διεθνώς, σε αντιδιαστολή με άλλες διαλέκτους όπως η PL-SQL της Oracle που διαφέρουν περισσότερο από το ANSI standards.

CLR

Ο Microsoft SQL Server 2005 – 2008 περιλαμβάνει ένα module (δηλαδή μονάδα μέτρησης) που λέγεται SQL CLR μέσω του οποίου ενσωματώνει το .NET μέσα στον SQL Server.Με το SQL CLR, οι αποθηκευμένες διαδικασίες μπορούν να γραφτούν σε οποιαδήποτε γλώσσα .NET συμπεριλαμβανομένου C# και VB.NET και να δημιουργήσουν μια stored procedure (αποθηκευμένη διαδικασία). Αυτό σημαίνει ότι ο SQL Server έχει όλες τις βιβλιοθήκες και πλεονεκτήματα του .NET, αυτόχθονα μέσα στο περιβάλλον του, τα οποία μπορεί να τα καλέσει οποιαδήποτε στιγμή.

3.2.2) Πλατφόρμα ανάπτυξης λογισμικού

Μερικές από τις πλατφόρμες για της ανάπτυξη ενός λογισμικού είναι το NetBeans,DEV++,Code block ,Eclipse κτλ.

Με βάση ότι η πτυχιακή απαιτεί χρήση java και λόγο ότι (Αφού έχει κατασκευαστεί από την Oracle, έχει υποστήριξη για της νεότερες εκδόσεις της Java νωρίτερα από κάθε άλλο IDE.) ακόμα είναι πιο φιλικοί ως προς τους νέους χρήστες επέλεξα το NetBeans

Λίγα λόγια για το NetBeans :
Το NetBeans είναι ένα επιτυχημένο ερευνητικό έργο ανοιχτής πηγής (open source) με μεγάλο αριθμό χρηστών, μια αναπτυσσόμενη κοινωνία, κοντά στους 100 (και πλέον!) συνεργάτες παγκοσμίως. Η Sun Microsystems ίδρυσε το ερευνητικό έργο ανοιχτής πηγής NetBeans τον Ιούνιο του 2000 και συνεχίζει να είναι ο κύριος ανάδοχος. Σήμερα δύο ερευνητικά έργα υπάρχουν: Το NetBeans IDE και το NetBeans Platform. Το NetBeans IDE είναι ένα περιβαλλοντικό ανάπτυγμα IDE - ένα εργαλείο στους προγραμματιστές για να γράψουν, να κάνουν compile, debug και να αναπτύξουν προγράμματα. Είναι γραμμένο σε Java - αλλά μπορεί να υποστηρίξει όλες τις γλώσσες προγραμματισμού. Υπάρχει επίσης ένας μεγάλος αριθμός υπομονάδων (modules) που βοηθάνε στην επέκταση της λειτουργικότητας του NetBeans IDE. Το NetBeans IDE είναι ένα ελεύθερο προϊόν δίχως περιορισμούς στον τρόπο χρησιμοποίησής του. Διαθέσιμο επίσης είναι το NetBeans Platform; ένα εκτατό θεμέλιο αποτελούμενο από υπομονάδες (modular) που χρησιμοποιείται σαν βάση λογισμικού για τη δημιουργία μεγάλων επιτραπέζιων (desktop) εφαρμογών. Οι ISV συνεργάτες διαθέτουν προσθήκες, επιπρόσθετα προγράμματα (plug-ins) που εύκολα συνενώνονται στο Platform και μπορούν επίσης να χρησιμοποιηθούν για την ανάπτυξη άλλων εργαλείων και λύσεων. Και τα δύο τα προϊόντα είναι ανοιχτής πηγής (open source) και ελεύθερα για εμπορική ή μη χρήση. Ο κώδικας πηγής (source code) είναι διαθέσιμος για επαναχρησιμοποίηση κάτω από το Common Development and Distribution License (CD)

3.3) Απαίτησης εργαλείων ανάπτυξης λογισμικού ως προς το σύστημα

Έχοντας επιλέξει τον sqlserver kai to NetBeans οι απαίτησης από της δυο πλατφόρμες ως προς το σύστημα μου είναι

3.3.1) NETBEANS

Ελάχιστες απαιτήσεις:

- **Microsoft Windows XP SP3/Vista SP1/ 7**,10**:**
  - **Processor:** 800MHz Intel Pentium III η άλλον ίσο
  - **Memory:** 512 MB
  - **Disk space:** 750 MB ελεύθερος χώρος

- **Ubuntu 9.10:**
  - **Processor:** 800MHz Intel Pentium III η άλλον ίσο
  - **Memory:** 512 MB
  - **Disk space:** 650 MB ελεύθερος χώρος

- **Solaris OS version 11 Express (SPARC):**
  - **Processor:** UltraSPARC II 450 MHz
  - **Memory:** 512 MB
  - **Disk space:** 650 MB ελεύθερος χώρος

- **Solaris OS version 11 Express (x86/x64 Platform Edition):**
  - **Processor:** AMD Opteron 1200 Series 1.8 GHz
  - **Memory:** 512 MB
  - **Disk space:** 650 MB ελεύθερος χώρος

- **Macintosh OS X 10.5 Intel:**
  - **Processor:** Dual-Core Intel (32 or 64-bit)
  - **Memory:** 512 MB
  - **Disk space:** 650 MB ελεύθερος χώρος

Προτεινόμενες απαιτήσεις :

- **Microsoft Windows 7 /8- 8.1/10:**
  - **Processor:** Intel Core i5 η άλλον ίσο
  - **Memory:** 2 GB (32-bit), 4 GB (64-bit)
  - **Disk space:** 1.5 GB ελεύθερος χώρος
- **Ubuntutu 15.04:**
  - **Processor:** Intel Core i5 η άλλον ίσο
  - **Memory:** 2 GB (32-bit), 4 GB (64-bit)
  - **Disk space:** 1.5 GB ελεύθερος χώρος
- **OS X 10.10 Intel:**
  - **Processor:** Dual-Core Intel
  - **Memory:** 4 GB
  - **Disk space:** 1.5 GB ελεύθερος χώρος

3.3.2 SQLSERVER

Απαιτήσεις:

Hard Disk	SQL Server 2014 απαιτεί τουλάχιστον 6 GB ελεύθερου χώρου στον δίσκο. ο απαιτούμενος χώρος ποικίλη στον SQL Server 2014 ανάλογα με το της θα εγκαταστήσεις .
Drive	A DVD drive, αν εγκατασταθεί από δίσκο .
Monitor	SQL Server 2014 χρειάζεται Super-VGA (800x600 ή υψηλότερης ανάλυσης monitor.
Internet	Πρόσβαση στο διαδίκτυο (fees may apply).

Memory[1]	**Minimum:** Express Editions: 512 MB All other editions: 1 GB **Recommended:** Express Editions: 1 GB All other editions: At least 4 GB and should be increased as database size increases to ensure optimal performance.
Processor Speed	**Minimum:** x86 Processor: 1.0 GHz x64 Processor: 1.4 GHz **Recommended:** 2.0 GHz or faster
Processor Type	x64 Processor: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support x86 Processor: Pentium III-compatible processor or faster

Κεφάλαιο 4

Χρήση προγράμματος και τρόπος κατασκευής

4.1 Δυνατότητες του προγράμματος - Τρόπος κατασκευής

Σκοπός του προγράμματος είναι η λειτουργία ενός συστήματος μηχανογράφησης για Super Market . Με δυνατότητες αποθήκευσης, επεξεργασίας ,διαγράφεις των δεδομένων. ακόμα παρέχεται λειτουργία εκτυπώσεις των δεδομένων αλλά και αυτόματη ενημέρωση της βάσης δεδομένων σε οποιαδήποτε αλλαγή . Όλα τα παραπάνω εξυπηρετούν στα πλαίσια της ομαλότερης λειτουργίας της επιχειρήσεις.

4.2 Δυνατότητα διαφορετικών εισόδων μεταξύ των χρηστών .

Επιλογή σύνδεσης ως διαχειριστής είτε ως απλός χρήστης και έλεγχος εγκυρότητας εισόδου . Αυτό επιτυγχάνετε με τον παρακάτω κώδικα :

```
String password = PassWordField.getText();  \\ θέτω την μεταβλητή string με την εκάστοτε κάθε φορά τιμή που αναγράφεται στο πεδίο εισόδου

String username = UserNameField.getText(); \\θέτω την μεταβλητή string με την εκάστοτε κάθε φορά τιμή που αναγράφεται στο πεδίο εισόδου


\\ αν ο κώδικας μου περιέχει αυτά τα στοιχεία εισέρχεται στην if αλλιώς αν τα στοιχεία ταιριάζουν στην εισέρχεται στην else if αν η είσοδο μας δεν καλύπτετε από την συνθήκη τότε εισέρχεται στην else .

        if (password.contains ("Usercc!")&& username.contains("User")){
          UserNameField.setText("");
          PassWordField.setText("");
          close();
          welcom w= new welcom (); \\ ανοίγει το παράθυρο welcom
          w.setVisible (true); \\κάνει το παράθυρο ορατό
        } else if ((password.contains ("Admincc!")&& username.contains("Admin"))){
          UserNameField.setText("");
          PassWordField.setText("");
          close();
          Console c=new Console (); \\ανοίγει το παράθυρο Console
          c.setVisible(true); \\ κάνει το παράθυρο ορατό


} else {
 JOptionPane.showMessageDialog(null,"WRONG PASSWORD or USERNAME !\n click OK and try again ","Wrong Pass",JOptionPane.ERROR_MESSAGE);
\\ εμφανιζεται μήνυμα σφάλματος
            PassWordField.setText .setText (""); \\βαζει κενό στο πεδίο
            UserNameField.setText .setText (""); \\βάζει κενό στο πεδίο
 }
```

4.3 Σύνδεση ως διαχειριστής

Έχοντας συνδεθεί πλέον σαν διαχειριστής έχουμε πρόσβαση στης εξής επιλογές :

- Suppliers
- Products
- Orders
- Low Capacity
- Sales

4.4 Ανάλυση καρτέλας Suppliers

- Δυνατότητα προσθήκης ανανεώσεις και διαγράφεις ενός προμηθευτή στην βάση με ελέγχω διπλότυπης εγράφης και διαγράφης
- Πίνακα εμφανίσεις όλων των καταχωρημένων προμηθευτών
- Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προμηθευτή από τον πίνακα(για κάποιο update)
- Καθαρισμός πεδίων (για insert νέου προμηθευτή)
- Δημιουργία εγγράφου pdf –εκτύπωσης βάσης προμηθευτών

4.4.1 Κώδικας προσθήκης ανανεώσεις και διαγράφεις προμηθευτή

Insert button

\\ το παρακάτω query ελέγχει αν υπάρχουν κάποια πεδία για την αποφυγή διπλών εγγραφών και στην συνέχεια φιλτράρει τα στοιχεία από τα πεδία και τα καταχωρεί στην βάση στον εκάστοτε πίνακα.

- String query="IF NOT EXISTS (SELECT SupPhone,SupProductReference FROM Suppliers WHERE SupPhone = '" + SupPhone.getText()+" ' and SupProductReference= ' "+SupProductReference.getText() " ') INSERT INTO Suppliers(SupName,SupSurName,SupCountry,SupCity,SupAddress,SupPostCode,SupCertificate,SupProductReference,SupPhone,SupCompany)VALUES ('"+ SupName.getText()+" ','"+ SupSurName.getText()+" ','"+ SupCountry.getText()+" ','"+ SupCity.getText()+ " ','" + SupAddress.getText()+ " ','" + SupPostCode.getText()+" ','"+ SupCertificate.getText()+ " ','" + SupProductReference.getText()+ " ','" +SupPhone.getText()+ " ','" +SupCompany.getText()+" ') " ;
- executeSQLQuery(query,"Updated");

\\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

Update button

\\ το παρακάτω query κάνει ενημέρωση οποία εγγραφεί θέλουμε από την βάση. φιλτράροντας τα στοιχεία από τα πεδία και τα καταχωρεί στην βάση στον εκάστοτε πίνακα.

- String query =" UPDATE Suppliers SET SupName='"+ SupName.getText()+" ' , SupSurName= '"+ SupSurName.getText()+" ' , SupCountry= '"+ SupCountry.getText()+" ' , SupCity= '"+ SupCity.getText()+" ' , SupAddress= '"+ SupAddress.getText()+" ' , SupPostCode= '"+ SupPostCode.getText()+" ' , SupCertificate= '"+ SupCertificate.getText()+" ' , SupProductReference= '"+ SupProductReference.getText()+" ' ,SupPhone = '"+SupPhone.getText()+" ' ,SupCompany = '"+SupCompany.getText()+ " ' WHERE SupId="+ SupId.getText();

- executeSQLQuery(query,"Updated");

  \\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

Delete button

\\Θέτουμε μια ακέραια μεταβλητή <<p>> με αποτέλεσμα ένα αναδυόμενο παράθυρο ελέγχου διαγραφείς <<yes or no>>

- int p = JOptionPane.showConfirmDialog(null, "Do you really want to delete this row ? ","Delete",JOptionPane.YES_NO_OPTION);

  \\ αν η επιλογή μας είναι yes τότε το p παίρνει την τιμή 0 και εκτελεί το query διαγραφείς με βάση το id

  if (p==0){
  String query =" DELETE FROM Suppliers WHERE SupId =" SupId.getText();
- executeSQLQuery(query, "Deleted"); }

  \\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

4.4.2 Κώδικας εμφάνισης όλων των καταχωρημένων προμηθευτών

```
public void show_suppliers_in_Jtable(){
\\ καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της <<getuserList>>
        ArrayList<Update_del_insert> list= getuserList();
        DefaultTableModel model =(DefaultTableModel)jtable_Suppliers.getModel();
        Object[] row =new Object[11]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
        for(int i =0;i< list.size();i++)          \\ φιλτράρω με μια επαναληπτική διαδικασία   τα δεδομένα όσο το μέγεθος τις << list  >>
        {
           row[0] = list.get(i).getsupId();
           row[1] = list.get(i).getsupName();
           row[2] = list.get(i).getsupSurName();
           row[3] = list.get(i).getsupCountry();
           row[4] = list.get(i).getsupCity();
           row[5] = list.get(i).getsupAdrress();
           row[6] = list.get(i).getsupPostCode();
           row[7] = list.get(i).getsupCertificate();
           row[8] = list.get(i).getsupProductReference();
           row[9] = list.get(i).getsupPhone();
           row[10] = list.get(i).getsupCompany ();

          model.addRow(row); \\ προσθετη <<row >> στον πίνακα
          }
 }
```

4.4.3 Κώδικας Αυτόματη εισαγωγή στοιχείων στα πεδία

```
private void jtable_SuppliersMouseClicked(java.awt.event.MouseEvent evt) {


        // Display selected Row In JtextFields from table

       //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει  όλα τα πεδία της εγγραφής  και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση  το i [?] και το row[?]);

        int i =jtable_Suppliers.getSelectedRow();

        TableModel model =jtable_Suppliers.getModel();

       SupId.setText((String) model.getValueAt(i,0));

       SupName.setText(model.getValueAt(i,1).toString());

       SupSurName.setText(model.getValueAt(i,2).toString());

       SupCountry.setText(model.getValueAt(i,3).toString());

       SupCity.setText(model.getValueAt(i,4).toString());

       SupAddress.setText(model.getValueAt(i,5).toString());

       SupPostCode.setText(model.getValueAt(i,6).toString());

       SupCertificate.setText(model.getValueAt(i,7).toString());

       SupProductReference.setText(model.getValueAt(i,8).toString());

       SupPhone.setText(model.getValueAt(i,9).toString());

       SupCompany.setText(model.getValueAt(i,10).toString();

    }
```

4.4.4 Κώδικας Καθαρισμός πεδίων

```
private void Clear_TextsActionPerformed(java.awt.event.ActionEvent evt)
{
    \\ βάζει στο κάθε πεδίο <<Jfieldtext>> το “κενό”
    SupName.setText("");
    SupSurName.setText("");
    SupId.setText("");
    SupCountry.setText("");
    SupCity.setText("");
    SupAddress.setText("");
    SupPostCode.setText("");
    SupCertificate.setText("");
    SupProductReference.setText("");
    SupPhone.setText("");
    SupCompany.setText("");
  }
```

4.4.5 Δημιουργία εγγράφου pdf –εκτύπωσης βάσης προϊόντων

```
private void Print_SuppliersActionPerformed(java.awt.event.ActionEvent evt) {
\\ εμφάνιση κεφαλίδας με το μήνυμα που θέλουμε (“whatever”)
        MessageFormat header = new MessageFormat("Suppliers");
 \\ εμφάνιση στο τέλος της σελίδας το νούμερο της  (“whatever”)
      MessageFormat footer = new MessageFormat("Page{0,number,integer}");
      try {
            \\ εκτυπώνει όλων τον πίνακα και τον εμφανίζει με προσαρμογή στην σελίδα
        jTable_Suppliers.print(JTable.PrintMode.FIT_WIDTH, header, footer);

      } catch (java.awt.print.PrinterException e){
            // εμφάνιση μηνύματος σφάλματος
        System.err.format("Cannot print %$%n",e.getMessage());
      }}
```

4.5 Ανάλυση καρτέλας Products

- Δυνατότητα προσθήκης ανανεώσεις και διαγράφεις ενός προϊόντος στην βάση με ελέγχω διπλότυπης εγράφης και διαγράφης
- Πίνακα εμφανίσεις όλων των καταχωρημένων προϊόντων
- Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προϊόντος από τον πίνακα(για κάποιο update)
- Καθαρισμός πεδίων (για insert νέου προϊόντος)
- Μενού εναλλαγής καρτελών (Product categories , Capacity level)
- Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode
- Δημιουργία εγγράφου pdf –εκτύπωσης βάσης προϊόντων

4.5.1 Κώδικας προσθήκης ανανεώσεις και διαγράφεις προϊόντων

Insert button

\\ το παρακάτω query ελέγχει αν υπάρχουν κάποια πεδία για την αποφυγή διπλών εγγραφών και στην συνέχεια φιλτράρει τα στοιχεία από τα πεδία και τα καταχωρεί στην βάση στον εκάστοτε πίνακα.

- String query="IF NOT EXISTS (SELECT Pro_Id,Pro_Name FROM Products WHERE Pro_Id = '" +Pro_Id.getText()+ " ' and Pro_Name = '"+Pro_Name.getText() " ')INSERT INTO Products(Pro_Id,Pro_Name,Pro_Price,Pro_Quantity,Pro_Supplier_id,Pro_Tax,Cat_products,Pro_Capa_level)VALUES ('"+Pro_Id.getText()+" ','"+Pro_Name.getText()+" ','"+Pro_Price.getText()+" ','"+Pro_Quantity.getText()+ " ','" +Pro_Supplier_id.getText()+ " ','" +Pro_Tax.getText()+" ','"+Cat_productss.getText()+ " ','" +Prio_levell.getText()+ " ') " ;
  executeSQLQuery(query,"Inserted");
  \\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

Update button

\\ το παρακάτω query κάνει ενημέρωση οποία εγγραφεί θέλουμε από την βάση. φιλτράροντας τα στοιχεία από τα πεδία και τα καταχωρεί στην βάση στον εκάστοτε πίνακα.

- String query ="UPDATE Products SET Pro_Id ='"+Pro_Id.getText()+" ' ,Pro_Name = '"+Pro_Name.getText()+" ' ,Pro_Price = '"+Pro_Price.getText()+" ' ,Pro_Quantity = '"+Pro_Quantity.getText()+" ' ,Pro_Supplier_id = '"+Pro_Supplier_id.getText()+" ' ,Pro_Tax = '"+Pro_Tax.getText()+" ' ,Cat_products = '"+Cat_productss.getText()+" ' ,Pro_Capa_level = '"+Prio_levell.getText()+" ' WHERE Pro_Id ="+Pro_Id.getText();
  executeSQLQuery(query,"Updated");
  \\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

Delete button

\\Θέτουμε μια ακέραια μεταβλητή <<p>> με αποτέλεσμα ένα αναδυόμενο παράθυρο ελέγχου διαγραφείς <<yes or no>>

- int p = JOptionPane.showConfirmDialog(null, "Do you really want to delete this row ? ","Delete",JOptionPane.YES_NO_OPTION);

\\ αν η επιλογή μας είναι yes τότε το p παίρνει την τιμή 0 και εκτελεί το query διαγραφείς με βάση το id

if (p==0){
String query ="DELETE FROM Products WHERE Pro_Id ="+Pro_Id.getText();

- executeSQLQuery(query, "Deleted"); }

\\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

4.5.2 Κώδικας εμφανίσεις όλων των καταχωρημένων προϊόντων

```
    public void show_Products_in_Jtable(){
\\ καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << getproList >>
        ArrayList<Update_del_insert_products> list= getproList();
        DefaultTableModel model =(DefaultTableModel)jTable_Products.getModel();
        Object[] row =new Object[8]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
        for(int i =0;i< list.size();i++)   \\ φιλτράρω με μια επαναληπτική διαδικασία   τα
           δεδομένα όσο το μέγεθος τις << list  >>
        {

           row[0] = list.get(i).getpro_Id();
           row[1] = list.get(i).getpro_Name();
           row[2] = list.get(i).getpro_Price();
           row[3] = list.get(i).getpro_Quantity();
           row[4] = list.get(i).getpro_Supplier_id();
           row[5] = list.get(i).getpro_Tax();
           row[6] = list.get(i).getcat_productss();
           row[7] = list.get(i).getprio_levell();
           model.addRow(row); \\ προσθετη <<row >> στον πίνακα

        }
         model.fireTableDataChanged();
    }
```

4.5.3 Κώδικας Αυτόματη εισαγωγή πεδίων

```
private void jTable_ProductsMouseClicked (java.awt. event.MouseEvent evt) {

    // Display selected Row In JtextFields from table

   //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει  όλα τα πεδία της εγγραφής και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση  το i [?] και το row[?]);


    int i =jTable_Products.getSelectedRow();

    TableModel model =jTable_Products.getModel();

   Pro_Id.setText((String) model.getValueAt(i,0));

   Pro_Name.setText(model.getValueAt(i,1).toString());

   Pro_Price.setText(model.getValueAt(i,2).toString());

   Pro_Quantity.setText(model.getValueAt(i,3).toString());

   Pro_Supplier_id.setText(model.getValueAt(i,4).toString());

   Pro_Tax.setText(model.getValueAt(i,5).toString());

   Cat_productss.setText(model.getValueAt(i,6).toString());

   Prio_levell.setText(model.getValueAt(i,7).toString());

 }
```

4.5.4 Κώδικας Καθαρισμός πεδίων

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        \\ βάζει στο κάθε πεδίο <<Jfieldtext>> το "κενό"
    Pro_Id.setText("");
    Pro_Name.setText("");
    Pro_Price.setText("");
    Pro_Quantity.setText("");
    Pro_Supplier_id.setText("");
    Pro_Tax.setText("");
    txt_serch.setText("");
    Cat_productss.setText("");
    Prio_levell.setText("");
```

4.5.6 Μενού εναλλαγής καρτελών

```
jTabbedPane1 = new javax.swing.JTabbedPane();
```

Δημιουργεί ένα νέο JTabbedPane στο υπαρχουν jTabbedPane

4.5.7 Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode

```
private void txt_serchKeyReleased(java.awt.event.KeyEvent evt) {
          //
      Connection con =getConnection(); \\ σύνδεση με βάση
            String query ="SELECT * FROM Products where Pro_Id =? "; \\εμφανισε
όλα τα πεδία  από τα προϊόντα με βάση το <<Pro_id>>
            Statement st;
            PreparedStatement pst = null;
            ResultSet rs;
           try{
             st = con.createStatement();
            pst=con.prepareStatement(query);
            pst.setString(1, txt_serch.getText()); \\ παίρνει το id του προϊόντος
            rs=pst.executeQuery();
// φιλτράρει τα στοιχεία του προϊόντος  με βάση το id του και τα εμφανίζει στα
εκάστοτε πεδία κάθε φορά που μπαίνει στην συνθήκη <<if>>
            if(rs.next()){
               String add1=rs.getString("Pro_Id");
               Pro_Id.setText(add1);
               String add2=rs.getString("Pro_Name");
               Pro_Name.setText(add2);
               String add3=rs.getString("Pro_Price");
               Pro_Price.setText(add3);
               String add4=rs.getString("Pro_Quantity");
               Pro_Quantity.setText(add4);
               String add5=rs.getString("Pro_Supplier_id");
               Pro_Supplier_id.setText(add5);
               String add6=rs.getString("Pro_Tax");
               Pro_Tax.setText(add6);
                String add7=rs.getString("Cat_products");
               Cat_productss.setText(add7);
               String add8=rs.getString("Pro_Capa_level");
               Prio_levell.setText(add8);
            }
} catch(Exception e)
      { JOptionPane.showMessageDialog(null, e); }
\\ μήνυμα σε περίπτωση σφάλματος
}
```

4.5.8 Δημιουργία εγγράφου pdf –εκτύπωσης βάσης προϊόντων

```
private void Print_ProductsActionPerformed(java.awt.event.ActionEvent evt) {
        \\ εμφάνιση κεφαλίδας με το μήνυμα που θέλουμε (“whatever”)
      MessageFormat header = new MessageFormat("Products");
      \\ εμφάνιση στο τέλος της σελίδας το νούμερο της  (“whatever”)
      MessageFormat footer = new MessageFormat("Page{0,number,integer}");
      try {
            \\ εκτυπώνει όλων τον πίνακα και τον εμφανίζει με προσαρμογή στην σελίδα

         jTable_Products.print(JTable.PrintMode.FIT_WIDTH, header, footer);

      } catch (java.awt.print.PrinterException e){
            // εμφάνιση μηνύματος σφάλματος
         System.err.format("Cannot print %$%n",e.getMessage());
      }
   }
```

4.6 Ανάλυση καρτέλας Orders

- Δυνατότητα προσθήκης μιας παραγγελίας στην βάση και άμεση ανανέωση της διαθέσιμης ποσότητας
- Πίνακα εμφανίσεις όλων των καταχωρημένων παραγγελιών – των προϊόντων
- Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προϊόντος από τον πίνακα
- Καθαρισμός πεδίων (για insert νέας παραγγελίας)
- Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode
- Δημιουργία εγγράφου pdf –εκτύπωσης βάσης παραγγελιών
- Αποστολη εμαιλ

4.6.1 Δυνατότητα προσθήκης μιας παραγγελίας στην βάση και άμεση ανανέωση της διαθέσιμης ποσότητας

Insert button

\\ το παρακατω query φιλτράρει τα στοιχεία από τα πεδία και τα καταχωρεί στην βάση στον εκάστοτε πίνακα

- String query="INSERT INTO Orders(Pro_Id,Pro_Name,Order_Quantity,Order_Date)VALUES ('"+Pro_Id.getText()+" ','"+Pro_Name.getText()+" ','"+Order_Quantity.getText()+" ',current_timestamp) ";
  executeSQLQuery(query,"Inserted");

\\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

\\ στην συνέχεια τρέχει το παρακάτω qquery το οποίο κάνει ενημέρωση στον πίνακα μας με βάση την παραγγελία μας και προσθετή στα πίνακα των προϊόντων σε έναν υπάρχον πεδίο μας τον αριθμό των προϊόντων που παραγγείλαμε

- String qquery= " UPDATE Products SET Pro_Quantity= Products.Pro_Quantity + Orders.Order_Quantity FROM Products INNER JOIN Orders ON Products.Pro_Id= Orders.Pro_Id where Order_Id = (SELECT MAX(Order_Id) FROM Orders)" ;
  executeSQLQuery(qquery,"updated");

\\ καλεί την συνάρτηση και αν κάποια τιμή έχει αλλαχτεί κάνει refresh τον πίνακα μας .

4.6.2 Πίνακα εμφανίσεις όλων των καταχωρημένων παραγγελιών – προϊόντων

```
public void show_Orders_in_Jtable(){
\\καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << getorderList >>
        ArrayList<Make_Order_Insert> list= getorderList();
        DefaultTableModel model =(DefaultTableModel)jTable_Orders.getModel();
        Object[] row =new Object[5]; \\θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
      for(int i =0;i< list.size();i++)  \\φιλτράρω με μια επαναληπτική διαδικασία  τα
        δεδομένα όσο το μέγεθος τις << list  >>


        {
           row[4] = list.get(i).getpro_Id();
           row[3] = list.get(i).getpro_Name();
           row[2] = list.get(i).getorder_Quantity();
           row[1] = list.get(i).getorder_Date();
           row[0] = list.get(i).getorder_Id();
       model.addRow(row);  \\ προσθετη <<row >> στον πίνακα
        }
    }
```

```
public void show_Products_in_Jtable(){
\\καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << getproList >>
        ArrayList<Update_del_insert_products> list= getproList();
        DefaultTableModel model =(DefaultTableModel)jTable_Prordes.getModel();
        Object[] row =new Object[2]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
      for(int i =0;i< list.size();i++) \\ φιλτράρω με μια επαναληπτική διαδικασία  τα
       δεδομένα όσο το μέγεθος τις << list  >>


        {
           row[0] = list.get(i).getpro_Id();
           row[1] = list.get(i).getpro_Name();
           model.addRow(row); \\ προσθετη <<row >> στον πίνακα
        }
      }
```

4.6.3 Κώδικας Αυτόματη εισαγωγή πεδίων

```
// Display selected Row In JtextFields from table
    //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει όλα τα πεδία της εγγραφής και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση to i [?] και το row[?]);

private void jTable_OrdersMouseClicked(java.awt.event.MouseEvent evt) {
    int i =jTable_Orders.getSelectedRow();
     TableModel model =jTable_Orders.getModel();
    Pro_Id.setText((String) model.getValueAt(i,0));
    Pro_Name.setText(model.getValueAt(i,1).toString());
    Order_Quantity.setText(model.getValueAt(i,2).toString());
    Order_Date.setText(model.getValueAt(i,3).toString());
    Order_Id.setText(model.getValueAt(i,4).toString());
  }
// Display selected Row In JtextFields from table
    //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει όλα τα πεδία της εγγραφής και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση to i [?] και το row[?]);

private void jTable_ProrDesMouseClicked(java.awt.event.MouseEvent evt) {
    int i =jTable_Prordes.getSelectedRow();
     TableModel model =jTable_Prordes.getModel();
    Pro_Id.setText((String) model.getValueAt(i,0));
    Pro_Name.setText(model.getValueAt(i,1).toString());
  }
```

4.6.4 Κώδικας Καθαρισμός πεδίων

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      \\ Βαζει στο κάθε πεδιο <<jfieldtext>> το <<”κενο”
    Pro_Id.setText("");
    Pro_Name.setText("");
    Order_Quantity.setText("");
    Order_Date.setText("");
    Order_Id.setText("");
  }
```

4.6.5 Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode

```
private void txt_serchKeyReleased(java.awt.event.KeyEvent evt) {
        //SERCH IN TABLE WITH BARCODE AS KEY AND SHOW AS
RESULT THE THE TABLE CONTECT ITEM
      Connection con =getConnection(); \\ Σύνδεση με βάση
            String query ="SELECT * FROM Products where Pro_Id =? ";  \\
εμδανισε όλα τα πεδια αππο τα προιοντα με βαση το <<Pro_Id>>
            Statement st;
            PreparedStatement pst = null;
            ResultSet rs;
           try{
             st = con.createStatement();
            pst=con.prepareStatement(query);
            pst.setString(1, txt_serch.getText()); \\ παίρνει το id του προϊόντος
            rs=pst.executeQuery();
// φιλτράρει τα στοιχεία του προϊόντος  με βάση το id του και τα εμφανίζει στα
εκάστοτε πεδία κάθε φορά που μπαίνει στην συνθήκη <<if>>

            if(rs.next()){
               String add1=rs.getString("Pro_Id");
               Pro_Id.setText(add1);
               String add2=rs.getString("Pro_Name");
               Pro_Name.setText(add2);
               String add3=rs.getString("Pro_Price");
               Pro_Price.setText(add3);
               String add4=rs.getString("Pro_Quantity");
               Pro_Quantity.setText(add4);
               String add5=rs.getString("Pro_Supplier_id");
               Pro_Supplier_id.setText(add5);
               String add6=rs.getString("Pro_Tax");
               Pro_Tax.setText(add6);
                String add7=rs.getString("Cat_products");
               Cat_productss.setText(add7);
               String add8=rs.getString("Pro_Capa_level");
               Prio_levell.setText(add8);
            }
} catch(Exception e)
      { JOptionPane.showMessageDialog(null, e); }
\\ μήνυμα σε περίπτωση σφάλματος
}
```

4.6.6 Δημιουργία εγγράφου pdf –εκτύπωσης βάσης παραγγελιών

```
private void jButton2ActionPerformed (java.awt.event.ActionEvent evt) {
        \\ εμφάνιση κεφαλίδας με το μήνυμα που θέλουμε (“whatever”)

        MessageFormat header = new MessageFormat("Orders");
      \\ εμφάνιση στο τέλος της σελίδας το νούμερο της  (“whatever”)
      MessageFormat footer = new MessageFormat("Page{0,number,integer}");
      try {
            \\ εκτυπώνει όλων τον πίνακα και τον εμφανίζει με προσαρμογή στην
σελίδα

          jTable_Orders.print(JTable.PrintMode.FIT_WIDTH, header, footer);

      } catch (java.awt.print.PrinterException e){
            // εμφάνιση μηνύματος σφάλματος

          System.err.format("Cannot print %$%n",e.getMessage());
      }
 }
```

4.6.7 Αποστολή εμαιλ

```
private void MailsentActionPerformed(java.awt.event.ActionEvent evt) {
        String From=fromm.getText();
        String To=too.getText();
        String Subject=jTextField2.getText();
        String Text=text.getText();
        Properties props= new Properties();
        props.put("mail.smtp.host","smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port","465");
        props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth","true");
        props.put("mail.smtp.port","465");
        Session session=Session.getDefaultInstance(props,
           new javax.mail.Authenticator() {
              protected PasswordAuthentication getPasswordAuthentication(){
                 return new PasswordAuthentication("χχχχχχχχχ@gmail.com",
"χχχχχχχχχχχ");
              } });
           try{
              Message message=new MimeMessage(session);
              message.setFrom(new InternetAddress(From));
              message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(To));
              message.setSubject(Subject);
              message.setText(Text);
              Transport.send(message);
              JOptionPane.showMessageDialog(null, "message sent");


           }catch(Exception e){
             JOptionPane.showMessageDialog(null, e);}}
```

4.7 Ανάλυση καρτέλας Low Capacity

- Πίνακα εμφανίσεις όλων των προϊόντων όπου υπάρχει έλλειψη

4.7.1 κώδικας query

\\ το παρακάτω query εμφανίζει από όλα τα προϊόντα με βάση το επίπεδο ανάγκης που έχουμε θέση

```
String query =" SELECT * FROM Priority_lvl INNER JOIN Products ON Products.Pro_Capa_level= Priority_lvl.Prio_level WHERE Pro_Capa_level = Prio_level  AND Pro_Quantity<=Prio_Number" ;
```

4.7.2 Κώδικας εμφανίσεις όλων των προϊόντων με έλλειψη

```
public void show_Capacity_in_Jtable(){
        ArrayList<Update_del_insert_products> list= getproList(); \\ καταχωρώ σε έναν δυναμικό πίνακα <<list>> τα αποτελέσματα της << getproList >>
        DefaultTableModel model =(DefaultTableModel)jTable_Capacity.getModel();
        Object[] row =new Object[7]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
        for(int i =0;i< list.size();i++) \\ φιλτράρω με μια επαναληπτική διαδικασία τα δεδομένα όσο το μέγεθος τις << list >>
        {
            row[0] = list.get(i).getpro_Id();
            row[1] = list.get(i).getpro_Name();
            row[2] = list.get(i).getpro_Price();
            row[3] = list.get(i).getpro_Quantity();
            row[4] = list.get(i).getpro_Supplier_id();
            row[5] = list.get(i).getpro_Tax();
            row[6] = list.get(i).getprio_levell();
model.addRow(row);}} \\ προσθετη <<row >> στον πίνακα
```

4.8 Ανάλυση καρτέλας Sales

- Πίνακα εμφανίσεις όλων των πωλήσεων

4.8.1 Κώδικας εμφανίσεις όλων των πωλήσεων

```
   public void show_Basket_in_Jtable(){
\\ καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << 
getBasketList >>


       ArrayList<Temporarily_Basket> list= getBasketList();


       DefaultTableModel model 
=(DefaultTableModel)jTable_ProSales.getModel();
       Object[] row =new Object[5]; \\ θέτω ποσά πεδία θα εμφανιστούν στον 
πίνακα
        for(int i =0;i< list.size();i++)   \\ φιλτράρω με μια επαναληπτική διαδικασία 
τα
          δεδομένα όσο το μέγεθος τις << list  >>


       {
          row[0] = list.get(i).getpro_Id();
          row[1] = list.get(i).getpro_Name();
          row[2] = list.get(i).getsales_Quantity();
          row[3] = list.get(i).getpro_Price();
          row[4] = list.get(i).getsales_Date();
          model.addRow(row); \\ προσθετη <<row >> στον πίνακα
       }
    }
```

4.9 Σύνδεση ως χρήστης

Έχοντας συνδεθεί πλέον σαν χρήστης έχουμε πρόσβαση στης εξής επιλογές :

- Basket
- Prices

4.10 Ανάλυση καρτέλας Basket

- Δυνατότητα προσθήκης –αφαίρεσης προϊόντος από το καλάθι – αυτόματος καθαρισμός πεδίων (με insert νέου προϊόντος στο καλάθι)
- Πίνακα εμφανίσεις προϊόντων στο καλάθι
- Δημιουργία βάσης ιστορικού πωλήσεων , άμεση ανανέωση αποθέματος προϊόντος και άδειασμα καλαθιού
- Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προϊόντος από τον πίνακα
- Άθροισμα και εμφάνισης τιμής καλαθιού
- Δημιουργία εγγράφου pdf –εκτύπωσης απόδειξης

4.10.1 Δυνατότητα προσθήκης –αφαίρεσης προϊόντος από το καλάθι – αυτόματος καθαρισμός πεδίων

\\ Το παρακατω query καταχωρη στον πινακα receipt της τιμες των πεδιων

Και στην συνεχεια διαγραφει τα πεδια για εισοδο νεου προιοντος

```
private void additionActionPerformed(java.awt.event.ActionEvent evt) {

      String query="INSERT INTO
Receipt(Pro_Id,Pro_Name,Sales_Quantity,Pro_Price)VALUES
('"+Pro_Id.getText()+" ','"+Pro_Name.getText()+" ','"+Sales_Quantity.getText()+"
','"+Pro_Price.getText()+" ') ";

        executeSQLQuery(query,"Inserted");

         Pro_Id.setText("");

         Pro_Name.setText("");

         Pro_Price.setText("");

         Sales_Quantity.setText("");

         search.setText("");

   }
```

Το παρακάτω query διαγραφεί την εγγραφή με βάση το id στον πίνακα receipt

Αφού πατηθεί yes στον έλεγχο εγκυρότητας

Και στην συνέχεια διαγραφεί τα πεδία για είσοδο νέου προϊόντος

```
private void minusActionPerformed(java.awt.event.ActionEvent evt) {
    int p = JOptionPane.showConfirmDialog(null, "Do you really want to delete this row ? ","Delete",JOptionPane.YES_NO_OPTION);
        if (p==0){
    String query ="DELETE FROM Receipt WHERE Pro_Id ="+Pro_Id.getText();
      executeSQLQuery(query,"Deleted");}
       Pro_Id.setText("");
       Pro_Name.setText("");
       Pro_Price.setText("");
       Sales_Quantity.setText("");
       search.setText("");
  }
```

4.10.2 Πίνακας εμφανίσεις προϊόντων στο καλάθι

```
public void show_Basket_in_Jtable(){
\\ καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << getBasketList >>

        ArrayList<Temporarily_Basket> list= getBasketList();

        DefaultTableModel model =(DefaultTableModel)jTable_ProSales.getModel();
        Object[] row =new Object[4]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
        for(int i =0;i< list.size();i++) \\ φιλτράρω με μια επαναληπτική διαδικασία  τα
δεδομένα όσο το μέγεθος τις << list  >>

        {
           row[0] = list.get(i).getpro_Id();
           row[1] = list.get(i).getpro_Name();
           row[2] = list.get(i).getsales_Quantity();
           row[3] = list.get(i).getpro_Price();
           model.addRow(row);  \\ προσθετη <<row >> στον πίνακα
         }
    }
```

4.10.3 Δημιουργία βάσης ιστορικού πωλήσεων , άμεση ανανέωση αποθέματος προϊόντος και άδειασμα καλαθιού

```
\\ το παρακάτω query δημιουργεί ένα ιστορικό πωλήσεων αντιγράφοντας τα
περιεχόμενα του καλαθιού receipt  και αποθηκεύοντας τα στον πίνακα sales .

String query= "INSERT INTO Sales (Pro_Id
,Pro_Name,Sales_Quantity,Pro_Price,Sales_Date ) SELECT
Pro_Id,Pro_Name,Sales_Quantity ,Pro_Price,current_timestamp FROM Receipt";

        makesales(query,"Inserted");

\\ το παρακάτω query με βάσει το id του προϊόντος ενημερώνει τον πίνακα products
στην βάσει για τα προϊόντα που αγοράστηκαν και τα αφαιρεί.

     String qquery= " UPDATE Products SET Pro_Quantity= Products.Pro_Quantity -
Receipt.Sales_Quantity FROM Products INNER JOIN Receipt ON Products.Pro_Id =
Receipt.Pro_Id" ;

        makesales(qquery,"updated");

\\το παρακάτω query διαγραφεί από τον πίνακα receipt όλα τα δεδομένα του

String qqquery= "DELETE  FROM Receipt";

        makesales(qqquery,"basket is empty");

    }
```

4.10.4 Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προϊόντος από τον πίνακα

```
private void jTable_ProSalesMouseClicked(java.awt.event.MouseEvent evt) {

    // Display selected Row In JtextFields from table

      //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει  όλα τα πεδία της εγγραφής  και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση  to i [?] και το row[?]);

     int i =jTable_ProSales.getSelectedRow();

     TableModel model =jTable_ProSales.getModel();

    Pro_Id.setText((String) model.getValueAt(i,0));

    Pro_Name.setText(model.getValueAt(i,1).toString());

    Sales_Quantity.setText(model.getValueAt(i,2).toString());

    Pro_Price.setText(model.getValueAt(i,3).toString());

  }
```

4.10.5 Άθροισμα και εμφάνισης τιμής καλαθιού

```
public void getSum() {
        Connection con =getConnection();\\ κάνει την συνδέσει
        Statement st;
        PreparedStatement pst = null;
        ResultSet rs;
       try{
\\ το παρακατω query κανει την πραξει για τον υπολογισμο τις τιμης του καλαθιου
          String qquery=" SELECT sum((Receipt.Sales_Quantity *
Receipt.Pro_Price*Pro_Tax)+(Receipt.Sales_Quantity * Receipt.Pro_Price))  AS
TotalItemsOrdered FROM Products INNER JOIN Receipt ON Products.Pro_Id =
Receipt.Pro_Id ";
         pst=con.prepareStatement(qquery);
                    rs=pst.executeQuery();

        if(rs.next()){
            float TotalItemsOrdered = rs.getFloat("TotalItemsOrdered");
            float nvNum=TotalItemsOrdered; \\αποθηκευη το αποτελεσμα στην
μεταβλητη  nvNum
            sum_calc.setText(valueOf(nvNum));\\εμφανιζει το αποτελεσμα στο field
         }else{
           JOptionPane.showMessageDialog(null, "Not Found");
        }

    }catch (Exception e) { JOptionPane.showMessageDialog(null, e); }
}
```

4.11 Ανάλυση καρτέλας Prices

- Πίνακα εμφανίσεις όλων των καταχωρημένων προϊόντων
- Αυτόματη εισαγωγή πεδίων στα αντίστοιχα κελιά με το πάτημα ενός προϊόντος από τον πίνακα(για κάποιο update)
- Καθαρισμός πεδίων (για insert νέου προϊόντος)
- Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode

4.11.1 Κώδικας εμφανίσεις όλων των καταχωρημένων προϊόντων

```
    public void show_Products_in_Jtable(){
\\ καταχωρώ  σε έναν δυναμικό πίνακα <<list>>  τα αποτελέσματα της << getproList >>
        ArrayList<Update_del_insert_products> list= getproList();
        DefaultTableModel model =(DefaultTableModel)jTable_Products.getModel();
        Object[] row =new Object[8]; \\ θέτω ποσά πεδία θα εμφανιστούν στον πίνακα
        for(int i =0;i< list.size();i++)   \\ φιλτράρω με μια επαναληπτική διαδικασία   τα
           δεδομένα όσο το μέγεθος τις << list  >>
        {


           row[0] = list.get(i).getpro_Id();
           row[1] = list.get(i).getpro_Name();
           row[2] = list.get(i).getpro_Price();
           row[3] = list.get(i).getpro_Quantity();
           row[4] = list.get(i).getpro_Supplier_id();
           row[5] = list.get(i).getpro_Tax();
           row[6] = list.get(i).getcat_productss();
           row[7] = list.get(i).getprio_levell();
           model.addRow(row); \\ προσθετη <<row >> στον πίνακα


        }
         model.fireTableDataChanged();
    }
```

4.11.2 Κώδικας Αυτόματη εισαγωγή πεδίων

```
private void jTable_ProductsMouseClicked (java.awt. event.MouseEvent evt) {

    // Display selected Row In JtextFields from table

   //ανάλογα με το << i >> του εκάστοτε <<row>> φιλτράρει  όλα τα πεδία της εγγραφής και τα εμφανίζει στα αντίστοιχα πεδία με την εντολή model.getValueAt( με βάση  το i [?] και το row[?]);


    int i =jTable_Products.getSelectedRow();

    TableModel model =jTable_Products.getModel();

   Pro_Id.setText((String) model.getValueAt(i,0));

   Pro_Name.setText(model.getValueAt(i,1).toString());

   Pro_Price.setText(model.getValueAt(i,2).toString());

   Pro_Quantity.setText(model.getValueAt(i,3).toString());

   Pro_Supplier_id.setText(model.getValueAt(i,4).toString());

   Pro_Tax.setText(model.getValueAt(i,5).toString());

   Cat_productss.setText(model.getValueAt(i,6).toString());

   Prio_levell.setText(model.getValueAt(i,7).toString());

}
```

4.11.3 Κώδικας Καθαρισμός πεδίων

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        \\ βάζει στο κάθε πεδίο <<Jfieldtext>> το “κενό”
    Pro_Id.setText("");
    Pro_Name.setText("");
    Pro_Price.setText("");
    Pro_Quantity.setText("");
    Pro_Supplier_id.setText("");
    Pro_Tax.setText("");
    txt_serch.setText("");
    Cat_productss.setText("");
    Prio_levell.setText("");
```

4.11.4 Αναζήτηση και εμφανίσει προϊόντος με βάσει το barcode

```
private void txt_serchKeyReleased(java.awt.event.KeyEvent evt) {
          //
      Connection con =getConnection(); \\ σύνδεση με βάση
            String query ="SELECT * FROM Products where Pro_Id =? "; \\εμφανισε
όλα τα πεδία  από τα προϊόντα με βάση το <<Pro_id>>
            Statement st;
            PreparedStatement pst = null;
            ResultSet rs;
           try{
             st = con.createStatement();
            pst=con.prepareStatement(query);
            pst.setString(1, txt_serch.getText()); \\ παίρνει το id του προϊόντος
            rs=pst.executeQuery();
// φιλτράρει τα στοιχεία του προϊόντος  με βάση το id του και τα εμφανίζει στα
εκάστοτε πεδία κάθε φορά που μπαίνει στην συνθήκη <<if>>
            if(rs.next()){
               String add1=rs.getString("Pro_Id");
               Pro_Id.setText(add1);
               String add2=rs.getString("Pro_Name");
               Pro_Name.setText(add2);
               String add3=rs.getString("Pro_Price");
               Pro_Price.setText(add3);
               String add4=rs.getString("Pro_Quantity");
               Pro_Quantity.setText(add4);
               String add5=rs.getString("Pro_Supplier_id");
               Pro_Supplier_id.setText(add5);
               String add6=rs.getString("Pro_Tax");
               Pro_Tax.setText(add6);
                String add7=rs.getString("Cat_products");
               Cat_productss.setText(add7);
               String add8=rs.getString("Pro_Capa_level");
               Prio_levell.setText(add8);
            }
} catch(Exception e)
      { JOptionPane.showMessageDialog(null, e); }
\\ μήνυμα σε περίπτωση σφάλματος
}
```

Κεφάλαιο 5
Συμπεράσματα

Αρχικά στην παρών εργασία αναπτύξαμε και αναλύσιμα τα βήματα από την σύλληψη τις ιδέας τον τρόπο ανάπτυξης που πρέπει να ακολουθήσουμε την σχεδίαση της ,μέχρι και την πρακτική υλοποίηση του λογισμικού μας .Έχοντας λάβει υπόψη ότι έχει αναφερθεί στα παραπάνω κεφάλαια είναι πλέον ξεκάθαρη η ανάγκη προγραμμάτων μηχανογράφησης για την διευκόλυνση της κάθε επιχείρησης .Διότι μέσω από αυτά δημιουργείτε ένα πιο συγκροτημένο περιβάλλον αποθήκευσης τον δεδομένων που εξυπηρετούν της ανάγκες τις εκάστοτε επιχείρησης .Με λίγα λογία είναι προσαρμοσμένα στης ανάγκες τους εξολοκλήρου . Τέλος η παρών εργασία ήταν ιδιαίτερα ενδιαφέρουσα διότι σε ένταση στο κλήμα ενός προγραμματιστή δίνοντας σου τα απαραίτητα εναύσματα.

Κεφάλαιο 6
Βιβλιογραφία

Τεχνολογία λογισμικού Μανόλης Γιακουμάκης, Νίκος Διαμαντίδης

Τεχνολογία Λογισμικού Ι ΒΑΣΙΛΕΙΟΣ ΒΕΣΚΟΥΚΗΣ

Η γλώσσα προγραμματισμού Java-OCP Συγγραφέας: Πανάγος, Νίκος, πληροφορικός

Java 2 platform Laura Lemay & Rogers Cadenhead

Συστήματα διαχείρισης βάσεων δεδομένων Raghu Ramakrishnan, Johannes Gehrke

https://el.wikipedia.org/wiki/Microsoft_SQL_Server

https://netbeans.org/index_el.html